DGIWG - 113

DGIWG Profiles of ISO 19107 and GML realization

Document Identification:	STD-DP-09-109-ed1.0.0-Profiles_of_ISO_19107_and_GML_realization
Publication Date:	16 December 2010
Edition:	1.0.0
Edition Date:	16 December 2010
Responsible Party:	DGIWG
Audience:	Approved for public release
Abstract:	The document defines the spatial profiles of GML and documents the process used to derive these profiles from the DGIWG 2D and 3D profile of ISO 19107 spatial schema.
Copyright:	© Copyright DGIWG, some rights reserved - (CC) (By:) Attribution-

Contents

Figures

Tables

Introduction

ISO 19107 provides conceptual schemas for describing and manipulating the spatial characteristics of geographic features. This document describes a number of profiles of ISO 19107 in UML that are intended to define data interchange formats for use with two-, 2½- and three-dimensional data. Further it defines their realization as profiles of GML. The different profiles identify the necessary geometry and topology characteristics required to support military applications.

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

Topology deals with the characteristics of geometric objects that remain unchanged if the space in which they exist *(surface on which they are situated)* is deformed - for example, when geographic data is transformed from one coordinate system to another**.** Within the context of geographic information, and expressed mathematically, topology is commonly used to describe "the connectivity of an n-dimensional graph, a property that is invariant under continuous transformation of the graph"**.** Computational topology provides information about the connectivity of geometric primitives that can be derived from the underlying geometry. It is used to answer such questions as:

a) "does A lie within B", for example identify all the features which exist within a particular area.
b) "is C connected to D", for example highlight all the lines which form the border of a particular area.

A key characteristic of the 2D spatial schema is the existence of a surface or manifold, which is either explicit or implicit on which all geometric and topological primitives are situated. This surface may be either based on a coordinate reference system, such as a geodetic ellipsoid, or a map projection plane, or on an implicit or explicit digital description of the surface of the earth such as a digital terrain model. As a result of the existence of this underlying surface these 2D profiles support 2-D and 2½D models. A 2½D model is a 2D model with 3D coordinates. The complexes and relationships between primitives of this model are unchanged through projection onto the x and y plane. Such models may contain geometric surface primitives. In a 2½D model surface primitives are fully defined in x and y but their interiors are left undefined in z.

Three-dimensional data is more complex than two-dimensional data. Three-dimensional data has more geometric primitives and inter-primitive associations than two-dimensional data. Two-dimensional data has three geometric primitives – points, curves and surfaces – available to represent geometric data where three-dimensional data has four, adding solids. In two-dimensional data there are two boundary/coboundary relationships where three-dimensional data has three. Two-dimensional data has only one *isolated in* relationship where three-dimensional has three. Not only is the number of primitives and associations greater, but the complexity of the associations is also greater. Curves are organized in a circular sequence around a node in two-dimensions. In three-dimensions, the curves cobounding a node are no longer ordered. In two-dimensions a curve may be cobounded by at most two surfaces, a "left" and a "right" surface. In three-dimensions a curve is cobounded by any number of surfaces and the surfaces are organized in a circular sequence around the curve. In two-dimensions the surface is the highest dimensional primitive and has no cobounding primitive. In three-dimensions a surface cobounds at most two solids, a "top" and a "bottom" solid.

A standardised data exchange format is essential for distributing information in a service oriented and network centric architecture. The Geography Markup Language (GML) is considered as the lingua franca of geographic web services defined and standardised by the Open Geospatial Consortium (OGC) and the International Organization of Standardization (ISO). GML itself is an international standard ISO 19136:2007 *Geographic information -- Geography Markup Language (GML)***.** This document makes use of GML to implement the defined military profiles.

This document is divided into several parts dealing with geometry (Clause 6.2) and topology (Clause 6.10) and GML profiling (Clause 8). Chapter 7 helps users to better understand possible implementations of these profiles by giving examples for common military use cases. This comprises the body of the document. The geometry and topology sections parallel the division in ISO 19107. The profiles describe collections of geometric primitives (with, in some cases, corresponding topological primitives) that meet certain criteria. They form a matrix (Table 6.2) where one axis is of increasing primitive dimension and the other is of increasing boundary association complexity. There are several annexes that are intended to help guide the

user of this document. This is achieved with a collection of examples and discussions of certain issues pertaining to profiling.

1 Scope

Specify a collection of general purpose 2D, 2½D and 3D profiles of 19107 with one general profile and multiple sub-profiles. Document the specified profiles and add examples to the profile document to guide users in understanding respective profile usage within an application schema.

Develop DGIWG Profiles of the ISO standard 19136 GML, based on military requirements as for the geometry part expressed in the profiles defined of the ISO spatial schema. The document defines the spatial profiles of GML and documents the process used to derive these profiles from the DGIWG 2D and 3D profile of ISO 19107 spatial schema.

2 Conformance

The DGIWG Profiles of ISO 19107 require conformance to ISO 19107 conformance classes. As defined by ISO 19106 these profiles are at conformance class 1. This is satisfied when a profile is established as a pure subset of one or more of the ISO geographic information standards possibly together with other ISO standards [ISO 19106:2004]. The following table lists the different sections related to each DGIWG GML profile.

Level	2D.0d	2D.1d	2D.2d	3D.0d	3D.1d	3D.2d	3D.3d
L1	A.1.1.1	A.1.1.2	A.1.1.3	A.1.1.1	A.1.1.2	A.1.1.3	A.1.1.4
L2	A.1.1.1	A.1.1.2	A.1.1.3	A.1.1.1	A.1.1.2	A.1.1.3	A.1.1.4
L3	A.1.1.1	A.1.1.2	A.1.1.3	A.1.1.1	A.1.1.2	A.1.1.3	A.1.1.4
L4	A.1.1.1	A.2.1.1	A.2.1.2	A.1.1.1	A.2.1.1	A.2.1.2	A.2.1.3
L5	A.1.1.1	A.4.1.1	A.4.1.2	A.1.1.1	A.4.1.1	A.4.1.2	A.4.1.3
L6	A.1.1.1	A.4.1.1	A.4.1.2	A.1.1.1	A.4.1.1	A.4.1.2	A.4.1.3

Table 2.1– ISO 19107 Conformance Classes for the DGIWG Profiles

Clause 6 of this profile uses the Unified Modelling Language (UML) to present conceptual schemas for describing the spatial characteristics of geographic features.

The DGIWG GML Profiles of ISO 19107 require conformance to ISO 19136 conformance classes. Most of the schema components specified in ISO 19136 implement concepts defined in the ISO 19100 series of International Standards. In these cases, the conformance classes defined in ISO 19136 are based on the conformance classes defined in the corresponding standard. Mandatory requirements of ISO 19136 remain mandatory. Differences to the base standard (options) will be made explicit by a note in the appropriate clause. Conformance to the DGIWG GML Profiles requires conformance to the relevant ISO 19136 conformance classes.

Any software implementation claiming conformance to the DGIWG GML Profiles defined in this document shall reference the GML profile supported by the implementation. The GML profile shall pass the applicable mandatory test cases in Annex A – Abstract test suite.

DGIWG GML profiles provide geometric and topologic classes, as a subset of GML, to limit its complexity. When defining an application schema, a user can then choose a UML DGIWG profile corresponding to a specific need (see section 7 - Use cases). Since most of the time GML data are not limited to 0d (points), 1d (curves), 2d (surfaces) or 3d (solids) geometries, only twelve compliance levels for GML application schema and GML data have been defined. The following table presents those compliance levels defined for GML application schema and GML data (named Lx_yD, with x= the spatial dimension and y = the topological level). For each of this compliance level, abstract tests are described (Annex B – Abstract test suite for GML application schema and Annex C – Abstract test suites for GML data).

	Spatial dimension	2D			3D			
	Primitive dimension	0d	1d	2d	0d	1d	2d	3d
Topological level	L1	L1_2D			L1_3D			
	L2	L2_2D			L2_3D			
	L3	L3_2D			L3_3D			
	L4	L4_2D			L4_3D			
	L5	L5_2D			L5_3D			
	L6	L6_2D			L6_3D			

Table 2.2– DGIWG GML Compliance levels

NOTE A profile is derived from base standards so that by definition, conformance to a profile is conformance to the base standards from which it is derived. [ISO 19106 Profiles]

3 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of these profiles.

ISO 19103:2005 — Geographic information – Conceptual schema language

ISO 19106:2004 — Geographic information – Profiles

ISO 19107:2003 — Geographic information – Spatial schema

ISO 19136:2007 — Geographic information – Geography Markup Language (GML)

4 Terms and definitions

NOTE Generally the terms and definitions of the base standard ISO 19106:2004, ISO 19107:2003 and ISO 19136:2007 apply to this profile as well. For a better understanding of this document, the main terms and definitions are repeated.

4.1. 2½ dimension

2 dimensional geometry with an additional height value added to x,y coordinates.

4.2. abstract test suite – ats

abstract test module specifying all the requirements to be satisfied for conformance
[ISO 19105:2000]

4.3. application schema

conceptual schema for data required by one or more applications
[ISO 19101:2002]

4.4. boundary

set that represents the limit of an entity
[ISO 19107:2003]

4.5. class
description of a set of objects that share the same attributes, operations, methods, relationships, and semantics
[ISO 19103:2005]

4.6. closure
union of the interior and boundary of a topological or geometric object

4.7. coboundary
set of topological primitives of higher topological dimension associated with a particular topological object, such that this topological object is in each of their boundaries

4.8. composite curve
sequence of curves such that each curve (except the first) starts at the end point of the previous curve in the sequence

4.9. computational geometry
manipulation of and calculations with geometric representations for the implementation of geometric operations

EXAMPLE Computational geometry operations include testing for geometric inclusion or intersection, the calculation of convex hulls or buffer zones, or the finding of shortest distances between geometric objects.

4.10. computational topology
topological concepts, structures and algebra that aid, enhance or define operations on topological objects usually performed in computational geometry

4.11. connected
property of a geometric object implying that any two direct positions on the object can be placed on a curve that remains totally within the object

4.12. connected node
node that starts or ends one or more edges

4.13. coordinate
one of a sequence of numbers designating the position of a point in N-dimensional space

NOTE In a coordinate reference system, the numbers must be qualified by units.

4.14. coordinate dimension [ISO 19111]
number of measurements or axes needed to describe a position in a coordinate system

4.15. coordinate reference system [ISO 19111]
coordinate system that is related to the real world by a datum

4.16. coordinate system
set of (mathematical) rules for specifying how coordinates are to be assigned to points

4.17. curve
1-dimensional geometric primitive, representing the continuous image of a line

NOTE The boundary of a curve is the set of points at either end of the curve. If the curve is a cycle, the two ends are identical, and the curve (if topologically closed) is considered to not have a boundary. The first point is called the start point, and the last is the end point. Connectivity of the curve is guaranteed by the "continuous image of a line" clause. A topological theorem states that a continuous image of a connected set is connected.

4.18. curve segment

1-dimensional geometric object used to represent a continuous component of a curve using homogeneous interpolation and definition methods

NOTE The geometric set represented by a single curve segment is equivalent to a curve.

4.19. cycle

<geometry>

spatial object without a boundary

NOTE Cycles are used to describe boundary components (see ring). A cycle has no boundary because it closes on itself, but it is bounded (i.e. it does not have infinite extent). A circle, for example, has no boundary, but is bounded.

4.20. direct position

position described by a single set of coordinates within a coordinate reference system

4.21. directed edge

directed topological object that represents an association between an edge and one of its orientations

NOTE A directed edge that is in agreement with the orientation of the edge has a + orientation, otherwise, it has the opposite (-) orientation. Directed edge is used in topology to distinguish the right side (-) from the left side (+) of the same edge and the start node (-) and end node (+) of the same edge and in computational topology to represent these concepts.

4.22. directed face

directed topological object that represents an association between a face and one of its orientations

NOTE The orientation of the directed edges that compose the exterior boundary of a directed face will appear positive from the direction of this vector; the orientation of a directed face that bounds a topological solid will point away from the topological solid. Adjacent solids would use different orientations for their shared boundary, consistent with the same sort of association between adjacent faces and their shared edges. Directed faces are used in the coboundary relation to maintain the spatial association between face and edge.

4.23. directed node

directed topological object that represents an association between a node and one of its orientations

NOTE Directed nodes are used in the coboundary relation to maintain the spatial association between edge and node. The orientation of a node is with respect to an edge, "+" for end node, "-" for start node. This is consistent with the vector notion of "result = end - start".

4.24. directed solid

directed topological object that represents an association between a topological solid and one of its orientations

NOTE Directed solids are used in the coboundary relation to maintain the spatial association between face and topological solid. The orientation of a solid is with respect to a face, "+" if the upNormal is outward, "-" if inward. This is consistent with the concept of "up = outward" for a surface bounding a solid.

4.25. directed topological object
topological object that represents a logical association between a topological primitive and one of its orientations

4.26. domain
well-defined set
NOTE Domains are used to define the domain and range of operators and functions.

4.27. edge
1-dimensional topological primitive
NOTE The geometric realization of an edge is a curve. The boundary of an edge is the set of one or two nodes associated to the edge within a topological complex.

4.28. edge-node graph
graph embedded within a topological complex composed of all of the edges and connected nodes within that complex.
NOTE The edge-node graph is a subcomplex of the complex within which it is embedded.

4.29. end node
node in the boundary of an edge that corresponds to the end point of that edge as a curve in any valid geometric realization of a topological complex in which the edge is used

4.30. end point
last point of a curve

4.31. exterior
difference between the universe and the closure
NOTE The concept of exterior is applicable to both topological and geometric complexes.

4.32. face
2-dimensional topological primitive
NOTE The geometric realization of a face is a surface. The boundary of a face is the set of directed edges within the same topological complex that are associated to the face via the boundary relations. These can be organized as rings.

4.33. feature
abstraction of real world phenomena
NOTE A feature may occur as a type or an instance. Feature type or feature instance should be used when only one is meant.

[ISO 19101:2002]

4.34. GML profile
subset of the GML schema
[ISO 19136:2007]

4.35. GML schema
schema components in the XML namespace "http://schema.opengis.net/gml/3.2" as specified in this International Standard
[ISO 19136:2007]

4.36. geometric aggregate
collection of geometric objects that has no internal structure

4.37. geometric boundary
boundary represented by a set of geometric primitives of smaller geometric dimension that limits the extent of a geometric object

4.38. geometric complex
set of disjoint geometric primitives where the boundary of each geometric primitive can be represented as the union of other geometric primitives of smaller dimension within the same set

NOTE : The geometric primitives in the set are disjoint in the sense that no direct position is interior to more than one geometric primitive. The set is closed under boundary operations, meaning that for each element in the geometric complex, there is a collection (also a geometric complex) of geometric primitives that represents the boundary of that element. Recall that the boundary of a point (the only 0D primitive object type in geometry) is empty. Thus, if the largest dimension geometric primitive is a solid (3D), the composition of the boundary operator in this definition terminates after at most three steps. It is also the case that the boundary of any object is a cycle.

[ISO 19107:2003]

4.39. geometric dimension
largest number n such that each direct position in a geometric set can be associated with a subset that has the direct position in its interior and is similar (isomorphic) to R n, Euclidean n-space

4.40. geometric object
spatial object representing a geometric set

NOTE A geometric object consists of a geometric primitive, a collection of geometric primitives, or a geometric complex treated as a single entity. A geometric object may be the spatial representation of an object such as a feature or a significant part of a feature.

4.41. geometric primitive
geometric object representing a single, connected, homogeneous element of space

NOTE Geometric primitives are non-decomposed objects that present information about geometric configuration. They include points, curves, surfaces, and solids.

[ISO 19107:2003]

4.42. geometric realization
geometric complex whose geometric primitives are in a 1 to 1 correspondence to the topological primitives of a topological complex, such that the boundary relations in the two complexes agree

NOTE In such a realization the topological primitives are considered to represent the interiors of the corresponding geometric primitives. Composites are closed.

4.43. geometric set
set of direct positions

NOTE This set in most cases is infinite.

4.44. interior
set of all direct positions that are on a geometric object but which are not on its boundary

NOTE The interior of a topological object is the homomorphic image of the interior of any of its geometric realizations. This is not included as a definition because it follows from a theorem of topology.

4.45. isolated node
node not related to any edge

4.46. node
0-dimensional topological primitive
NOTE The boundary of a node is the empty set.

4.47. namespace <XML>
collection of names, identified by a URI reference, which are used in XML documents as element names and attribute names [W3C XML Namespaces]
[ISO 19136:2007]

4.48. planar topological complex
topological complex that has a geometric realization that can be embedded in Euclidean 2 space.

4.49. point
0-dimensional geometric primitive, representing a position
NOTE The boundary of a point is the empty set.

4.50. profile
set of one or more base standards or subsets of base standards, and, where applicable, the identification of chosen clauses, classes, options and parameters of those base standards, that are necessary for accomplishing a particular function
NOTE A profile is derived from base standards so that by definition, conformance to a profile is conformance to the base standards from which it is derived.

NOTE Sub-profile means that a hierarchical lower profile is incorporated in a hierarchical higher profile, e.g. L1.2D.0d in L1.2D.2d

[ISO 19106:2004]

4.51. proxy
item that stands in for or represents another item.

4.52. ring
simple curve which is a cycle
NOTE Rings are used to describe boundary components of surfaces in 2-D coordinate systems.

4.53. schema
formal description of a model
NOTE In general, a schema is an abstract representation of an object's characteristics and relationship to other objects. An XML schema represents the relationship between the attributes and elements of an XML object (for example, a document or a portion of a document)

[ISO 19101:2002]
[ISO 19136:2007]

4.54. schema <XML Schema>
collection of schema components within the same target namespace
EXAMPLE Schema components of W3C XML Schema are types, elements, attributes, groups, etc.

[ISO 19136:2007]

4.55. schema document <XML Schema>
XML document containing schema component definitions and declarations

NOTE The W3C XML Schema provides an XML interchange format for schema information. A single schema document provides descriptions of components associated with a single XML namespace, but several documents may describe components in the same schema, i.e. the same target namespace.

[ISO 19136:2007]

4.56. set
unordered collection of related items (objects or values) with no repetition

4.57. simple
property of a geometric object that its interior is isotropic (all points have isomorphic neighbourhoods), and hence everywhere locally isomorphic to an open subset of a Euclidean coordinate space of the appropriate dimension

NOTE This implies that no interior direct position is involved in a self-intersection of any kind.

4.58. spatial object
object used for representing a spatial characteristic of a feature

4.59. specialization

4.60. start node
node in the boundary of an edge that corresponds to the start point of that edge as a curve in a valid geometric realization of the topological complex in which the edge is used

4.61. start point
first point of a curve

4.62. subcomplex
complex all of whose elements are also in a larger complex

NOTE Since the definitions of geometric complex and topological complex require only that they be closed under boundary operations, the set of any primitives of a particular dimension and below is always a subcomplex of the original, larger complex. Thus, any full planar topological complex contains an edge-node graph as a subcomplex.

4.63. surface
2-dimensional geometric primitive, locally representing a continuous image of a region of a plane

NOTE The boundary of a surface is the set of oriented, closed curves that delineate the limits of the surface.

4.64. surface patch
2-dimensional, connected geometric object used to represent a continuous portion of a surface using homogeneous interpolation and definition methods

4.65. topological boundary
boundary represented by a set of oriented topological primitives of smaller topological dimension that limits the extent of a topological object

NOTE The boundary of a topological complex corresponds to the boundary of the geometric realization of the topological complex.

4.66. topological complex
collection of topological primitives that is closed under the boundary operations

NOTE Closed under the boundary operations means that if a topological primitive is in the topological complex, then its boundary objects are also in the topological complex.

[ISO 19107:2003]

4.67. topological dimension
minimum number of free variables needed to distinguish nearby direct positions within a geometric object from one another

4.68. topological object
spatial object representing spatial characteristics that are invariant under continuous transformations

NOTE A topological object is a topological primitive, a collection of topological primitives, or a topological complex.

[ISO 19107:2003]

4.69. topological primitive
topological object that represents a single, non-decomposable element

NOTE A topological primitive corresponds to the interior of a geometric primitive of the same dimension in a geometric realization.

[ISO 19107:2003]

4.70. UML application schema
application schema written in UML according to ISO 19109
[ISO 19136:2007]

4.71. Uniform Resource Identifier (URI)
unique identifier for a resource, structured in conformance with IETF RFC 2396

NOTE The general syntax is <scheme>::<scheme-specific-part>. The hierarchical syntax with a namespace is

<scheme>://<authority><path>?<query> - see [RFC 2396].

4.72. vector geometry
representation of geometry through the use of constructive geometric primitives

5 Symbols, and abbreviated terms

2D	2-Dimensional
2½D	2 dimensional geometry with an additional height value added to x,y coordinates.3D 3-Dimensional
DGIWG	Defence Geospatial Information Working Group
GML	Geography Markup Language NOTE The acronym GML which was previously used in ISO also as Generalized Markup Language (which led to SGML Standard Generalized Markup Language, ISO 8879).
ISO	International Organisation for Standardization
MGCP	Multinational Geospatial Co-production Program
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
UML	Unified Modelling Language
VMap	Vector Map
VRF	Vector Relational Format
WFS	OGC Web Feature Service
XML	eXtensible Markup Language

6 UML Profiles

6.1 Introduction

The DGIWG profiles of ISO 19107- Geographic Information- Spatial schema define profiles of geometry and topology to satisfy the requirements of the military community. This section defines 6 levels of profiles for 2D / 2½D and 3D, 37 profiles in all, 12 of which are documented in this section. Table 6.1 defines the individual levels. Each level has a number and a name (e.g. L1 – free geometry). Each level builds on the previous one, adding classes and constraints. Elements added to a level to get the next level are underlined in the level in which they were added.

	Constraints	Classes
L1 – free geometry	none	• Basic set of primitive classes.
L2 – no self intersection	• Primitives are simple, i.e. do not self intersect.	• Basic set of primitive classes.
L3 – primitives disjoint	• Primitives are simple and disjoint, i.e. do not self intersect and do not intersect each other.	• Primitives and proxies. • Proxies required for shared geometry.
L4 – complex	• Primitives are simple and disjoint, i.e. do not self intersect and do not intersect each other. • All boundary primitives exist. • Collection of primitives forms a complex.	• Primitives and proxies. • Proxies required for shared geometry and composites. • Complexes and composites.
L5 – full topology	• Primitives are simple and disjoint, i.e. do not self intersect and do not intersect each other. • All boundary primitives exist. • Collection of primitives forms a complex. • All boundary and coboundary relationships.	• Primitives and proxies. • Proxies required for shared geometry and composites. • Complexes and composites. • Boundary classes. • Topology classes.
L6 – partitioned space	• Primitives are simple and disjoint, i.e. do not self intersect and do not intersect each other. • All boundary primitives exist. • Collection of primitives forms a complex. • All boundary and coboundary relationships. • Space is partitioned.	• Primitives and proxies. • Proxies required for shared geometry and composites. • Complexes and composites. • Boundary classes. • Topology classes.

Table 6.1 – Profile levels

Table 6.2 categorizes the 37 profiles. The vertical categories are the 6 profile levels. The first 4 levels contain strictly GM_ classes where as the last 2 levels have both GM_ and TP_ classes. Horizontally the table has two major categories and 4 minor categories. The two major categories, 2D and 3D, specify the spatial dimension, that is the dimension of the space in which the geometric and topological objects are defined. The minor categories; 0d, 1d, 2d, and 3d; specify the maximal primitive dimension in the profile. The maximal primitive dimension cannot be greater than the spatial dimension. As a result the minor category 3d is only part of the major category 3D where as all the other minor categories are in both major categories. By combining the level, major, and minor category identification a profile designation is derived, e.g. L3.2D.1d designates the level 3, 2 dimensional profile with primitive dimension less than or equal to 1. In addition to these designations the documented profiles (dark yellow) have been given more descriptive names (e.g. L1.3D.3d is called 3D Free Geometry).

NOTE: 2½D geometry is included in the DGIWG 2D spatial schema profiles.

spatial dimension	2D			3D			
primitive dimension	0d	1d	2d	0d	1d	2d	3d
L1 – free geometry	**L1.2D.0d**	**L1.2D.1d**	**L1.2D.2d** → 2DFreeGeometry	**L1.3D.0d**	**L1.3D.1d**	**L1.3D.2d**	**L1.3D.3d** → 3DFreeGeometry
L2 – no self intersection	**L2.2D.0d** = L1.2D.0d	**L2.2D.1d**	**L2.2D.2d** → 2DSimple	**L2.3D.0d** = L1.2D.0d	**L2.3D.1d**	**L2.3D.2d**	**L2.3D.3d** → 3DSimple
L3 – primitives disjoint	**L3.2D.0d** *(~ L1.2D.0d)*	**L3.2D.1d**	**L3.2D.2d** → 2DShared	**L3.3D.0d** *(~ L1.2D.0d)*	**L3.3D.1d**	**L3.3D.2d**	**L3.3D.3d** → 3DShared
L4 – complex	**L4.2D.0d**	**L4.2D.1d**	**L4.2D.2d** → 2DComplex	**L4.3D.0d**	**L4.3D.1d**	**L4.3D.2d**	**L4.3D.3d** → 3DComplex
L5 – full topology	**L5.2D.0d**	**L5.2D.1d**	**L5.2D.2d** → 2DTopology	**L5.3D.0d**	**L5.3D.1d**	**L5.3D.2d**	**L5.3D.3d** → 3DTopology
L6 – partitioned space			**L6.2D.2d** → 2DTessellation				**L6.3D.3d** → 3DTessellation

Table 6.2 – Overview of profiles

Of the 37 profiles the maximal primitive dimensions of both spatial dimensions are documented for each level. There are 7 profiles at each level except for level 6. At this level there are only profiles where the spatial dimension and the maximal primitive dimension are the same. These profiles have a universe primitive in their datasets. There is a universe face for the 2D spatial dimension and there is a universe volume for the 3D spatial dimension.

All 37 profiles are fully specified in the “Profile Matrix.xls” Excel spreadsheet that is bundled within the same .zip folder. The spreadsheet has a number of tables, some normative, others informative, and yet others contain support data. The first table, “Levels”, corresponding to Table 6.1, defines the 6 different profile levels. The second table, “Overview”, corresponding to Table 6.2, gives an overview of the 37 profiles. The third table, “Profiles”, specifies the 37 profiles. The left most two columns, organized by their section in the ISO document, list all the classes in ISO 19107 with their properties. To the right the individual profiles are defined such that cells containing “X” indicate that that class or property is used, cells containing “*” indicate that the class or property is not used, and cells containing “S” followed by a number indicate a referenced specialization, where the referenced specialization modifies the class or property. The next table, “Specializations”, lists all the specializations used in the various profiles, assigning them each an identifier. This table also contains modifications to and errors in the standard – these are discussed in the two following paragraphs. The “Profile Class Lists” table lists the classes in each profile. The “Profile Specialization Lists” table lists the specializations in each profile. These last two tables are used to create this document and to generate the GML for the UML profiles. The rest of the tables are informative or contain support data.

The UML of the original ISO 19107 standard and the UML of the ISO/TC 211 harmonized model differ slightly. The discrepancies appear to be errors that were introduced in the harmonized model. E1: In the original 19107 the type GM_Aggregate is concrete, whereas in the harmonized model it is abstract. This appears unnecessary and the original standard is used. E2: In the original 19107 the type GM_Complex is concrete, where as in the harmonized model it is abstract. This is an error since the only collections of primitives that could form complexes would be composites, which function like primitives. This would prevent, for example, three bounded edges that share a common node from being represented with a GM_Complex, although they form a perfectly valid complex. E3: In the original 19107 the directed topological primitives TP_DirectedNode, TP_DirectedEdge, TP_DirectedFace, and TP_DirectedSolid are specified both as concrete and abstract. The harmonized model clears up this ambivalence by specifying all these types as abstract. This choice prevents the required proxy primitives from being instantiated. This

document assumes the directed topological primitives are concrete, specializing them to abstract in some profiles.

The profiles use a slightly modified version of the ISO 19107 standard. The two modifications are intended to simplify the model without losing any information. M1: A GM_Ring behaves like a GM_Curve and GM_Shell behaves like a GM_Surface, therefore these types inherit from GM_CompositeCurve and GM_CompositeSurface respectively. In the standard, composite primitives inherit from GM_Complex. This requires profiles that use the types GM_SurfaceBoundary and GM_SolidBoundary to also implement GM_Complex with its added complexity. In order to simplify the lower level profiles it was found desirable to exclude GM_Complex and derived classes. To circumvent this requirement, instead of inheriting from GM_CompositeCurve, GM_Ring realizes it and inherits directly from GM_Object. Similarly, GM_Shell, instead of inheriting from GM_CompositeSurface, realizes it and inherits directly from GM_Object. M2: ISO 19107 is inconsistent in how it deals with segmentation. GM_Curve and GM_Surface are defined by, and are required to have a defined interior. GM_Solid is not. GM_Solid is defined by its boundary and ISO 19107 offers no facility to define its interior. This works in a 3D space but does not make sense in a 2D or 1D space; a surface is fully defined by its boundary in a 2D space, and a curve is fully defined by its boundary in a 1D space. Since some of the profiles are defined in a 2D space the standard is modified so that it works well for both 2D and 3D profiles.

In this document the sections of the 12 documented profiles are each divided into 4 parts: Introduction, Class diagrams, Included constructs, and Specializations. The introduction gives a brief description of the profile. The class diagrams section contains class diagrams showing the classes of the profile with their specializations. The included constructs section lists the classes included in the profile by their section in ISO 19107 with references to their specializations. All other constructs are omitted. The specializations section lists the specializations of the classes of the profile. Each specialization has the classes it specializes as well as a specialization reference number in its title. The specialization reference number refers to the specialization in the profile matrix. Class specification text is presented in `Courier`. If a multiplicity is specialized, the original multiplicity is in parentheses tabbed to the right as in

```
GM_OrientableSurface::composite[0..1] : GM_Shell                    ([0..n])
```

where the multiplicity is restricted from [0..n] to [0..1]. For an overview of the use of UML in schema profiles, refer to ISO 19103 – Conceptual Schema Language. OCL (part of UML) is used to explicitly document all constraints.

6.2 2D Free Geometry (L1.2D.2d – 2D free geometry)

6.2.1 Introduction

This profile consists of an unconstrained collection of 2D geometric primitives. Free geometry refers to data which has been captured with no constraints. This data is often referred to as spaghetti data. No connectivity is established between geometries and geometries are allowed to self intersect. Thus, the primitives, as a collection, do not form a geometric complex, although subsets of the collection may coincidentally or expressly form complexes. Spaghetti data or free geometry can be useful however, if all that is required is a visual image or plot of a map and no spatial analysis is to be performed, e.g. free geometry can be represented by Shapefiles or VPF level 0.

6.2.2 Class diagrams

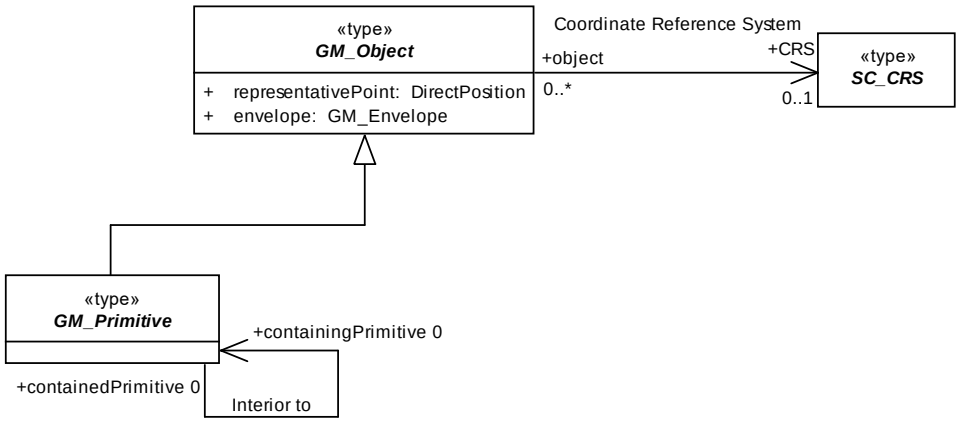


Figure 6.2.1 – Geometry object

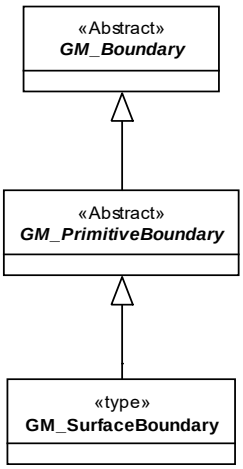


Figure 6.2.2 – Geometry boundary

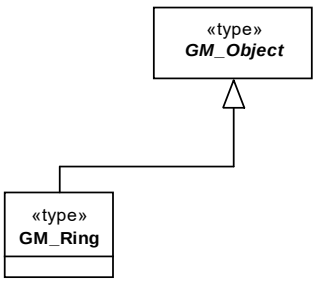


Figure 6.2.3 – Geometry boundary components

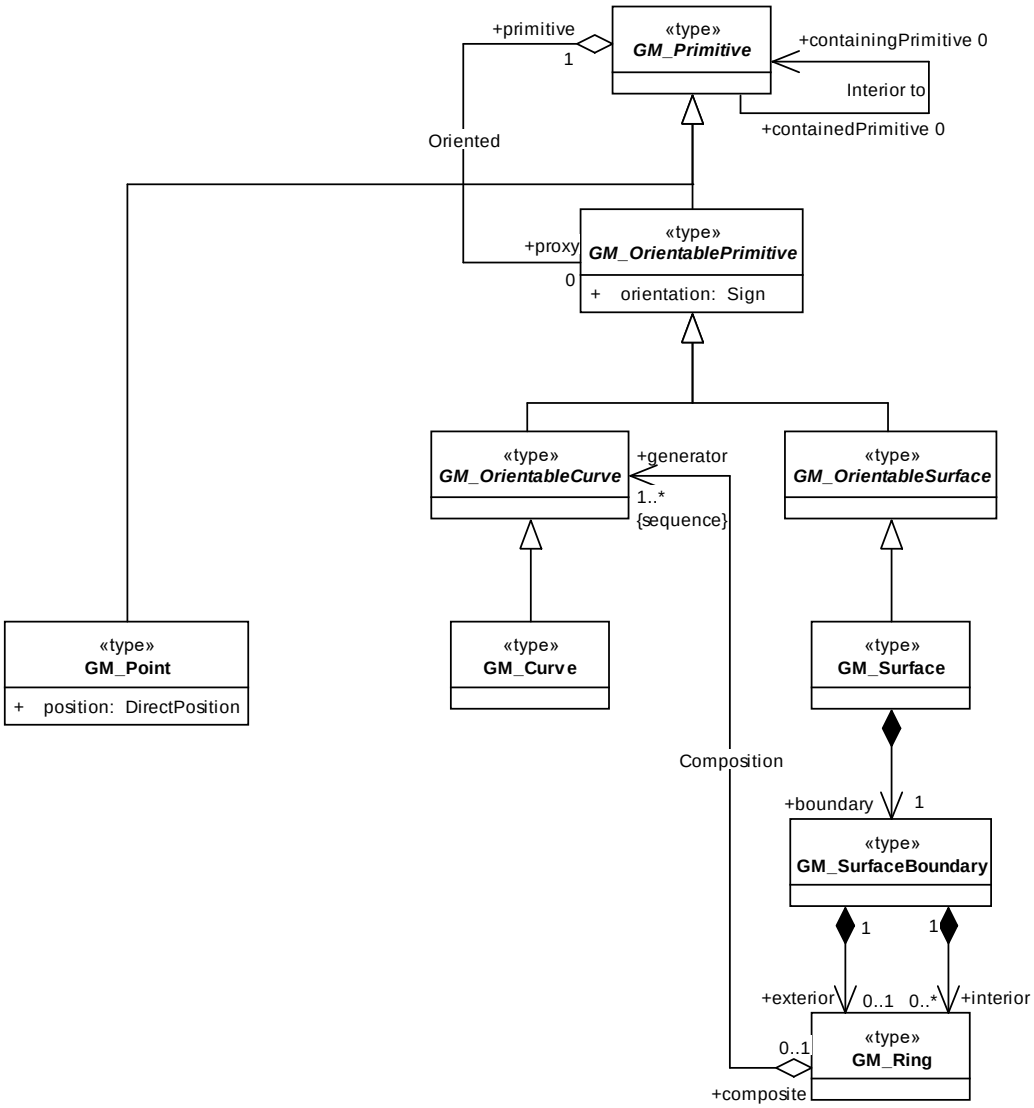


Figure 6.2.4 – Geometry primitive

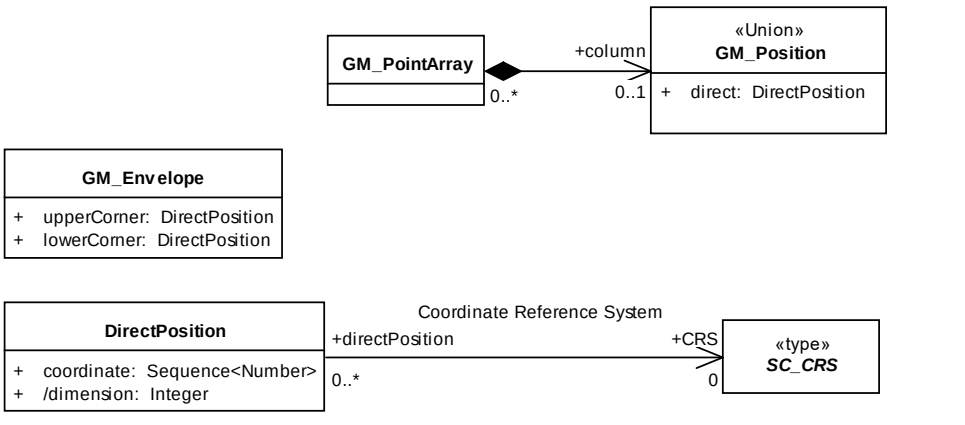


Figure 6.2.5 – Coordinate package

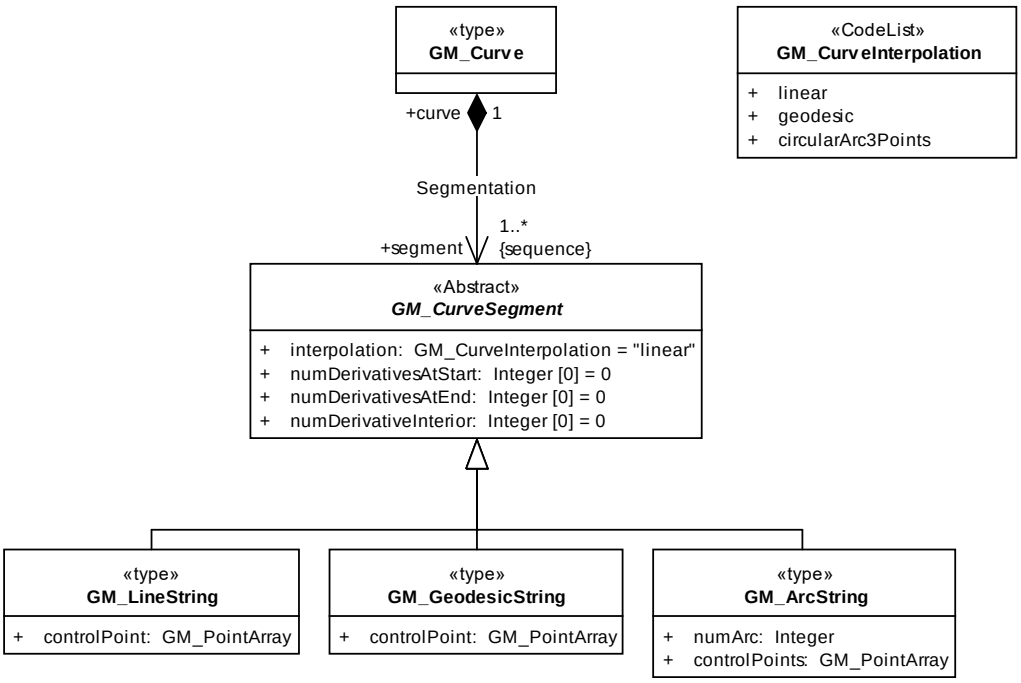


Figure 6.2.6 – Curve segment

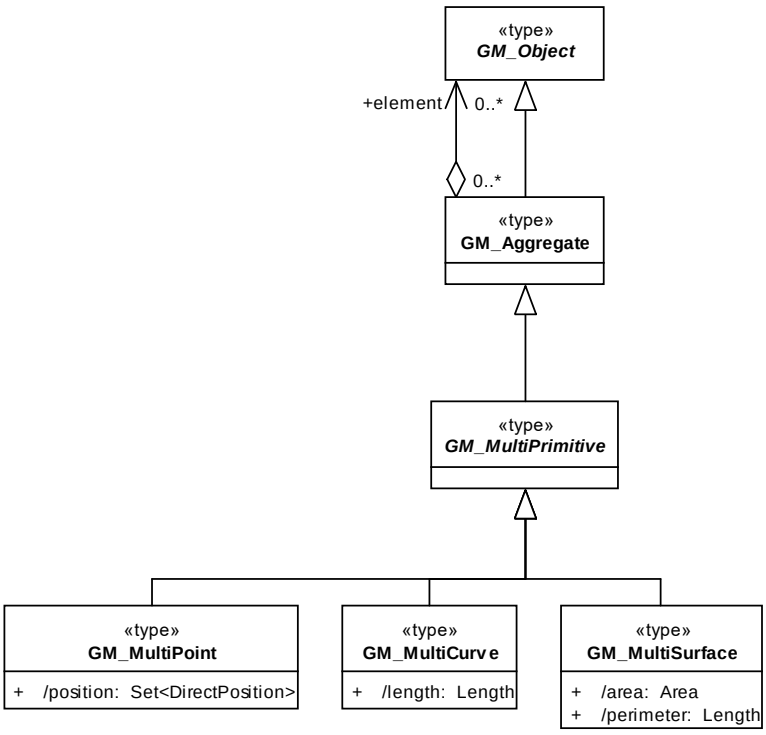


Figure 6.2.7 – Geometry aggregation

6.2.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.2.4.1(S1), 6.2.4.2(S2)

- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.10 GM_Primitive – specializations 6.2.4.3(S6), 6.2.4.4(S8)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.2.4.5(S9)
- 6.3.15 GM_OrientableSurface – specializations 6.2.4.6(S10), 6.2.4.7(S12)
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specializations 6.2.4.7(S12), 6.2.4.9(S14)
- 6.4.1 DirectPosition – specialization 6.2.4.10(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.2.4.11(S18)
- 6.4.6 GM_PointArray
- 6.4.8 GM_CurveInterpolation – specialization 6.2.4.12(S19)
- 6.4.9 GM_CurveSegment – specializations 6.2.4.8(S13), 6.2.4.13(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface

6.2.4 Specializations

6.2.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.2.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Object::envelope[0,1] : GM_Envelope
```

6.2.4.3 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.2.4.4 GM_Primitive (S8)

The Oriented association is not used. The multiplicity of the proxy role of the Oriented association between GM_Primitive and GM_OrientablePrimitive is constrained from [0,2] to [0].

```
GM_Primitive::proxy[0] : GM_OrientablePrimitive                        ([0,2])
```

6.2.4.5 GM_OrientableCurve (S9)

GM_OrientableCurve is abstract and not instantiable.

6.2.4.6 GM_OrientableSurface (S10)

GM_OrientableSurface is abstract and not instantiable.

6.2.4.7 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.2.4.8 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                    ([0,1])
```

6.2.4.9 GM_Surface (S14)

A GM_Surface is described by its boundary; GM_SurfacePatch is not used. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [0].

```
GM_Surface::patch[0] : GM_SurfacePatch                                  ([0..n])
```

6.2.4.10 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                   ([0,1])
```

6.2.4.11 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                                 ([0,1])
GM_Position::indirect[0] : GM_PointRef                                  ([0,1])
```

6.2.4.12 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.2.4.13 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                     ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                       ([0,1])
```

6.3 3D Free Geometry (L1.3D.3d – 3D free geometry)

6.3.1 Introduction

This profile consists of an unconstrained collection of 3D geometric primitives. Free geometry refers to data which has been captured with no constraints. This data is often referred to as spaghetti data. No connectivity is established between geometries and geometries are allowed to self intersect. Thus, the primitives, as a collection, do not form a geometric complex, although subsets of the collection may coincidentally or expressly form complexes. Spaghetti data or free geometry can be useful however, if all that is required is a visual image or plot of a map and no spatial analysis is to be performed. e.g. free geometry can be represented by 3D Shapefiles.

6.3.2 Class diagrams

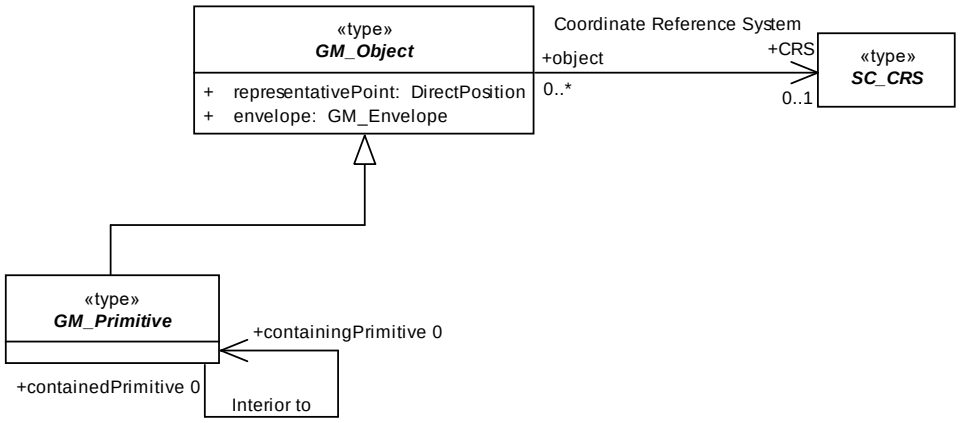


Figure 6.3.1 – Geometry object

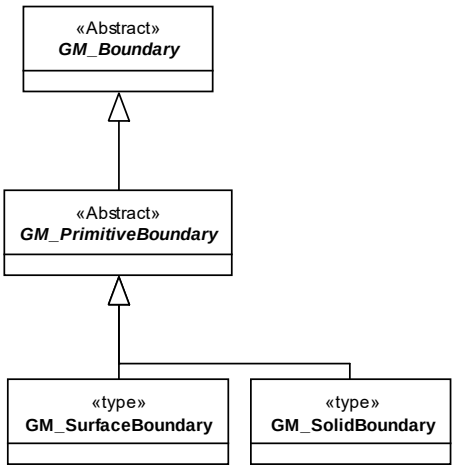


Figure 6.3.2 – Geometry boundary

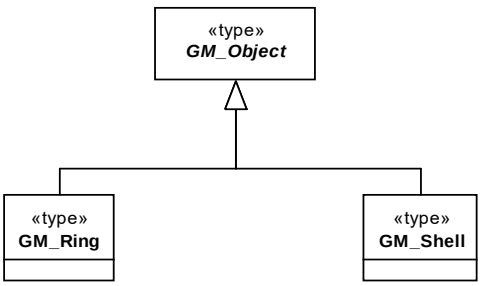


Figure 6.3.3 – Geometry boundary components

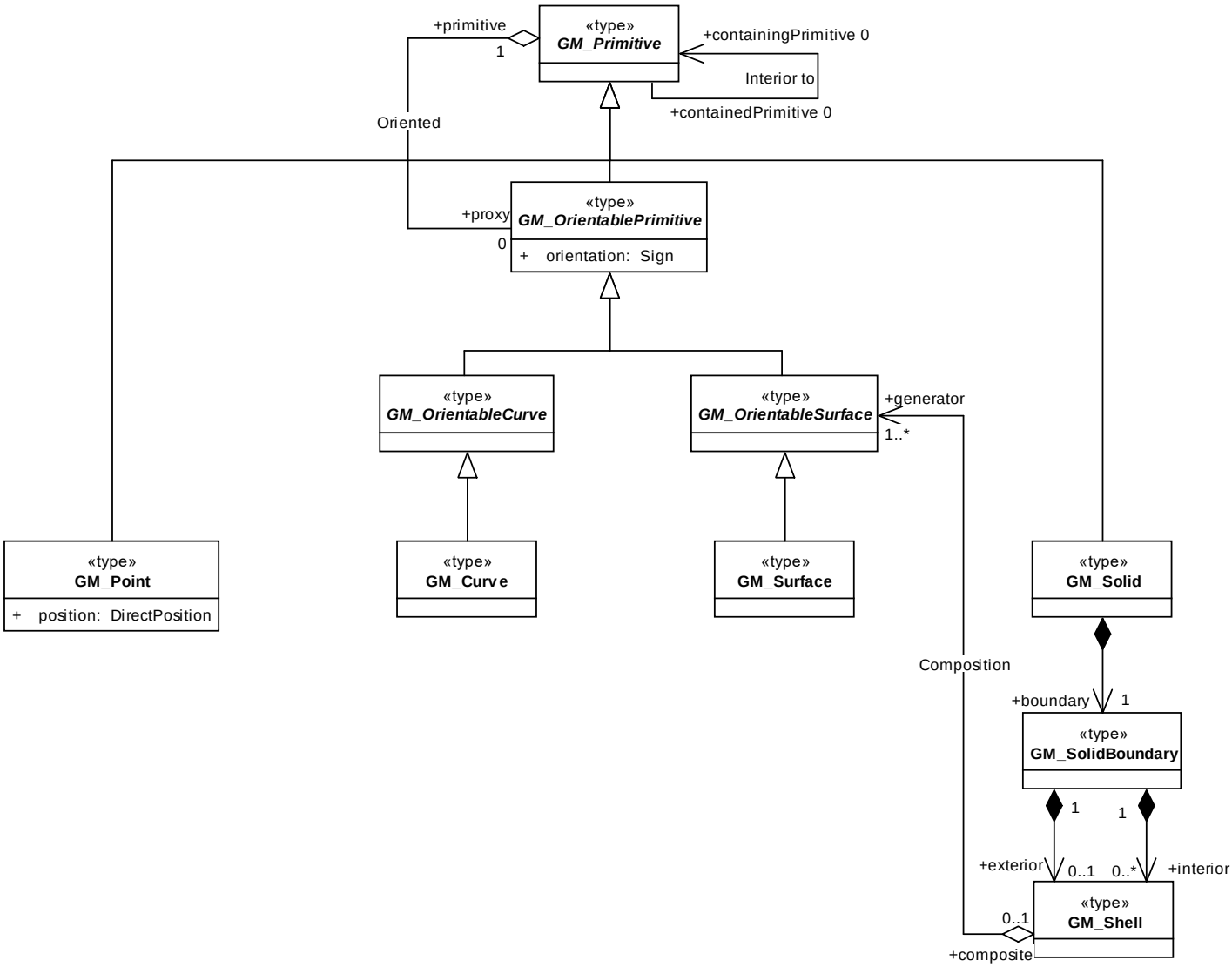


Figure 6.3.4 – Geometry primitive

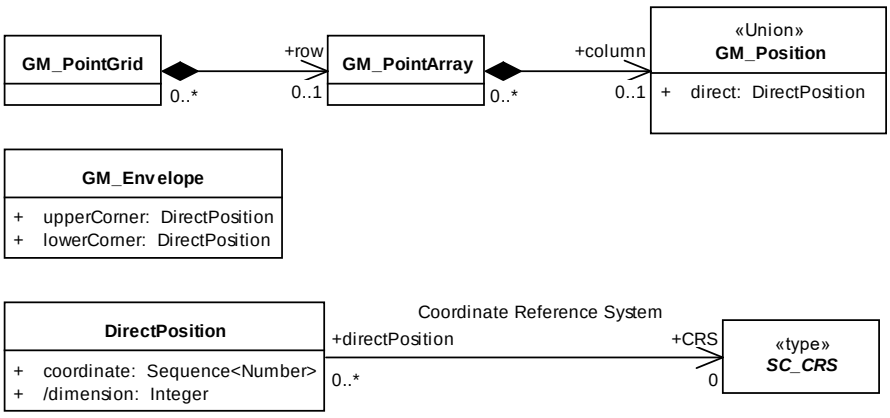


Figure 6.3.5 – Coordinate package

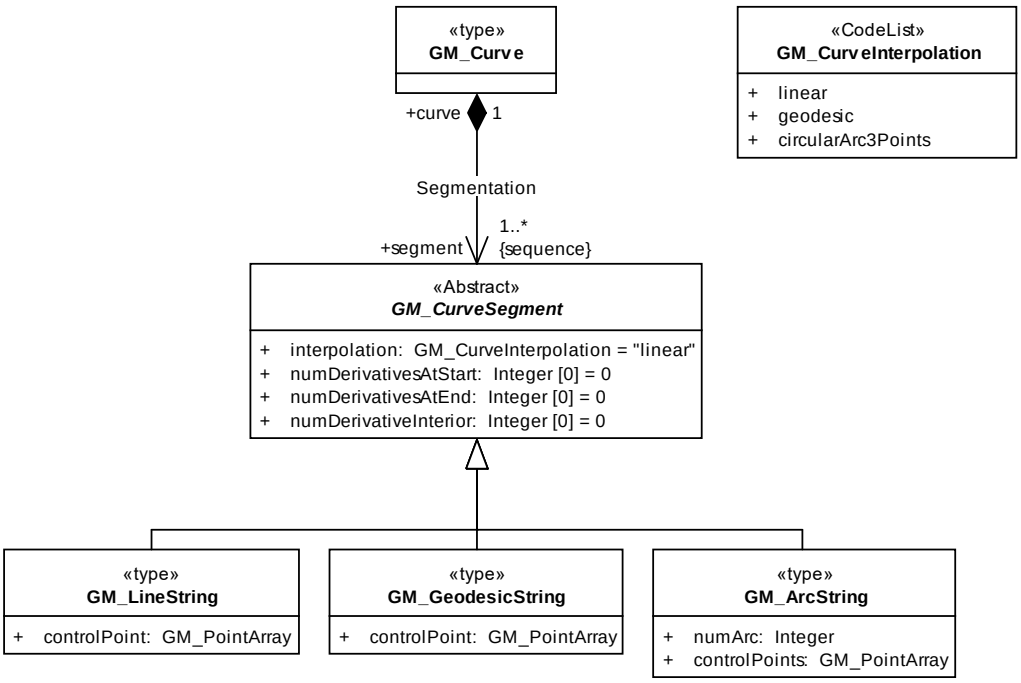


Figure 6.3.6 – Curve segment

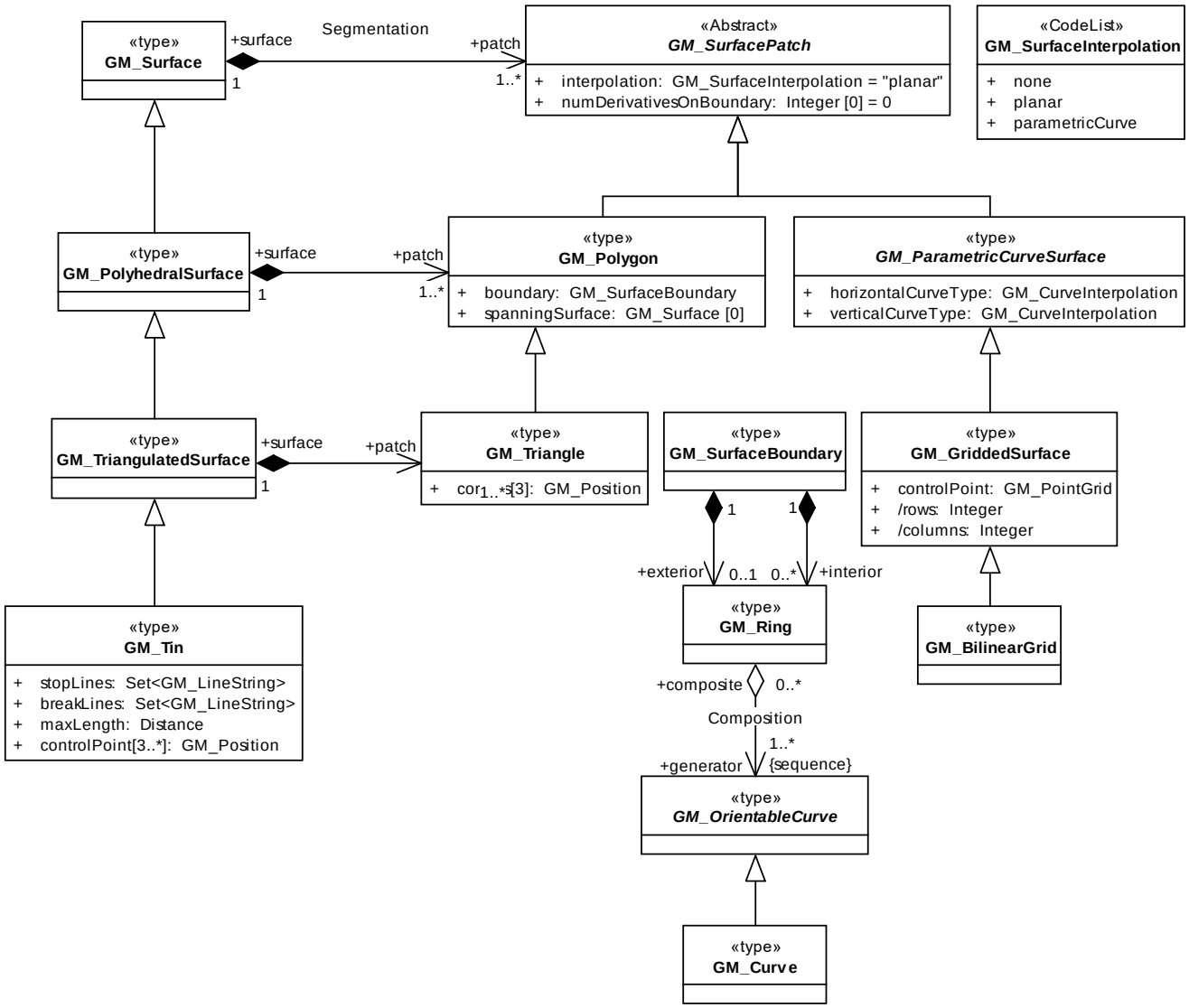


Figure 6.3.7 – Surface patch

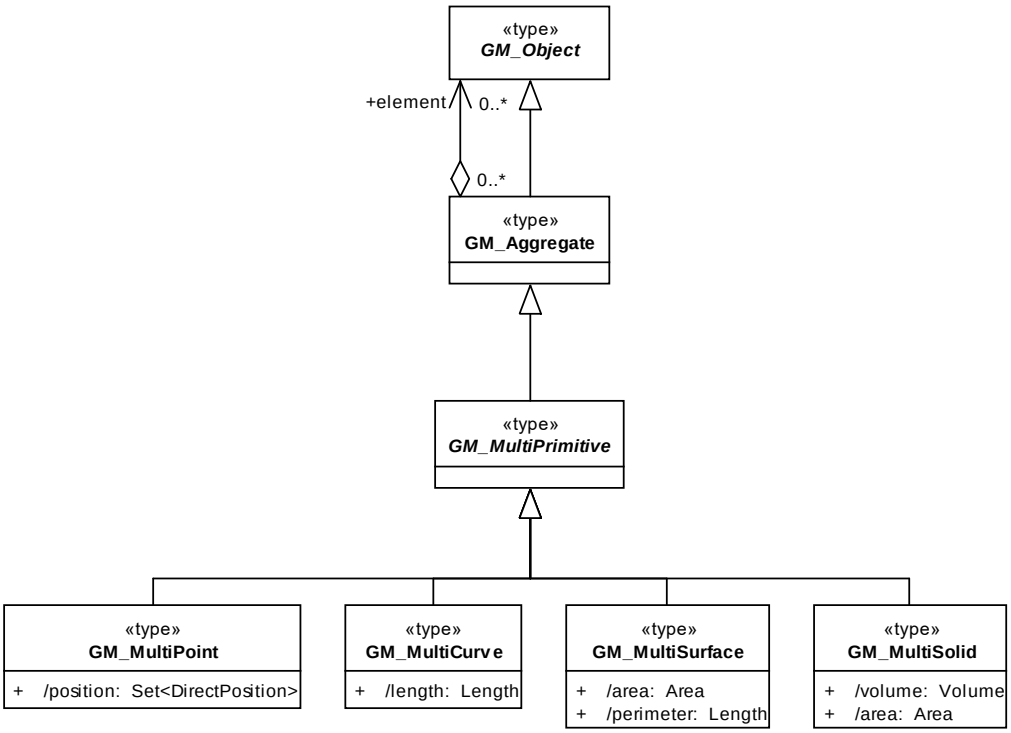


Figure 6.3.8 – Geometry aggregation

6.3.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.3.4.1(S1), 6.3.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring – specialization 6.3.4.3(S3)
- 6.3.7 GM_SurfaceBoundary – specialization 6.3.4.3(S3)
- 6.3.8 GM_Shell
- 6.3.9 GM_SolidBoundary
- 6.3.10 GM_Primitive – specializations 6.3.4.4(S6), 6.3.4.5(S8)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.3.4.6(S9)
- 6.3.15 GM_OrientableSurface – specialization 6.3.4.7(S10)
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specialization 6.3.4.9(S15)
- 6.3.18 GM_Solid – specialization 6.3.4.10(S16)
- 6.4.1 DirectPosition – specialization 6.3.4.11(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.3.4.12(S18)
- 6.4.6 GM_PointArray
- 6.4.6 GM_PointGrid

- 6.4.8 GM_CurveInterpolation – specialization 6.3.4.13(S19)
- 6.4.9 GM_CurveSegment – specializations 6.3.4.8(S13), 6.3.4.14(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.4.32 GM_SurfaceInterpolation – specialization 6.3.4.15(S21)
- 6.4.34 GM_SurfacePatch – specializations 6.3.4.16(S22), 6.3.4.17(S23)
- 6.4.35 GM_PolyhedralSurface
- 6.4.36 GM_Polygon – specialization 6.3.4.18(S24)
- 6.4.37 GM_TriangulatedSurface
- 6.4.38 GM_Triangle
- 6.4.39 GM_Tin
- 6.4.40 GM_ParametricCurveSurface
- 6.4.41 GM_GriddedSurface
- 6.4.45 GM_BilinearGrid
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.5.7 GM_MultiSolid

6.3.4 Specializations

6.3.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.3.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.3.4.3 GM_SurfaceBoundary, GM_Ring (S3)

GM_SurfaceBoundary and GM_Ring are only used to support GM_Polygon.

6.3.4.4 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.3.4.5 GM_Primitive (S8)

The Oriented association is not used. The multiplicity of the proxy role of the Oriented association between GM_Primitive and GM_OrientablePrimitive is constrained from [0,2] to [0].

```
GM_Primitive::proxy [0] : GM_OrientablePrimitive                       ([0,2])
```

6.3.4.6 GM_OrientableCurve (S9)

GM_OrientableCurve is abstract and not instantiable.

6.3.4.7 GM_OrientableSurface (S10)

GM_OrientableSurface is abstract and not instantiable.

6.3.4.8 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                    ([0,1])
```

6.3.4.9 GM_Surface (S15)

The interior of a GM_Surface patch is described by GM_SurfacePatches. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [1..n].

```
GM_Surface::patch[1..n] : GM_SurfacePatch                               ([0..n])
```

6.3.4.10 GM_Solid (S16)

The boundary operation of GM_Solid is changed to a composition relationship to GM_SolidBoundary.

```
GM_Solid::boundary[1] : GM_SolidBoundary
```

6.3.4.11 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                   ([0,1])
```

6.3.4.12 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                                 ([0,1])
GM_Position::indirect[0] : GM_PointRef                                  ([0,1])
```

6.3.4.13 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.3.4.14 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                     ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                       ([0,1])
```

6.3.4.15 GM_SurfaceInterpolation (S21)

The GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.

```
GM_SurfaceInterpolation::
none
planar
parametricCurve
```

6.3.4.16 GM_SurfacePatch (S22)

The multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained from [0,1] to [1].

```
GM_SurfacePatch::surface[1] : GM_Surface                                ([0,1])
```

6.3.4.17 GM_SurfacePatch (S23)

The multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained from [0,1] to [0].

```
GM_SurfacePatch::numDerivativesOnBoundary[0] : Integer                ([0,1])
```

6.3.4.18 GM_Polygon (S24)

The multiplicity of the spanningSurface attribute of GM_Polygon is constrained from [0..1] to [0].

```
GM_Polygon::spanningSurface[0] : GM_Surface                           ([0,1])
```

6.4 2D Simple (L2.2D.2d – 2D no self intersection)

6.4.1 Introduction

This profile consists of a collection of 2D non-self intersecting geometric primitives. Simple geometry refers to data which has been captured with limited constraints. No connectivity is established between geometries but geometries are not allowed to self intersect. Thus, the primitives, as a collection, do not form a geometric complex, although subsets of the collection may coincidentally or expressly form complexes. Simple geometry can be useful if all that is required is a visual image or plot of a map and no spatial analysis is to be performed and is the first step in cleaning up spaghetti data.

6.4.2 Class diagrams

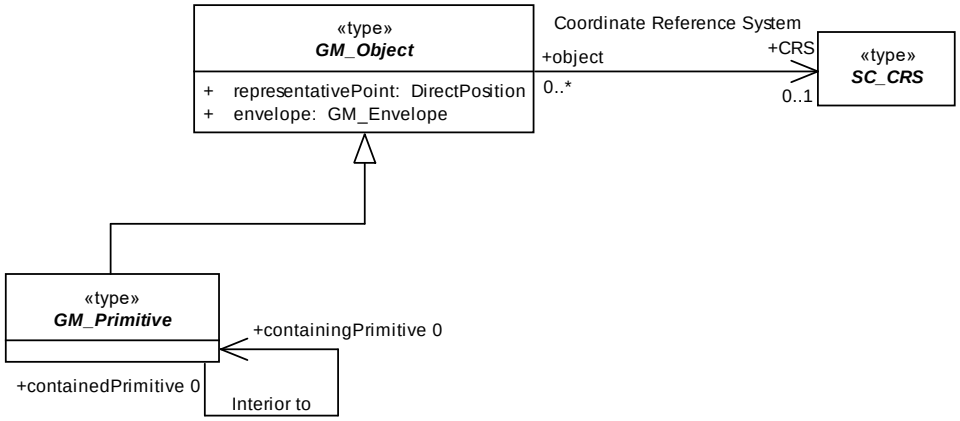


Figure 6.4.1 – Geometry object

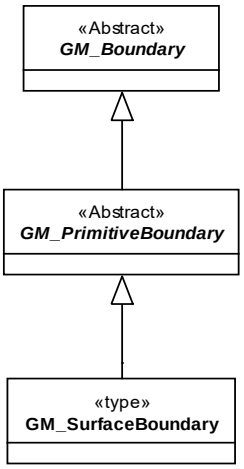


Figure 6.4.2 – Geometry boundary

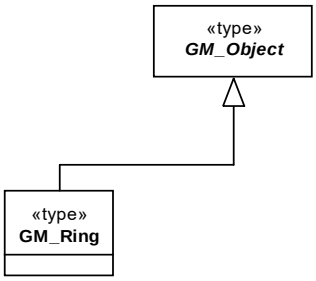


Figure 6.4.3 – Geometry boundary components

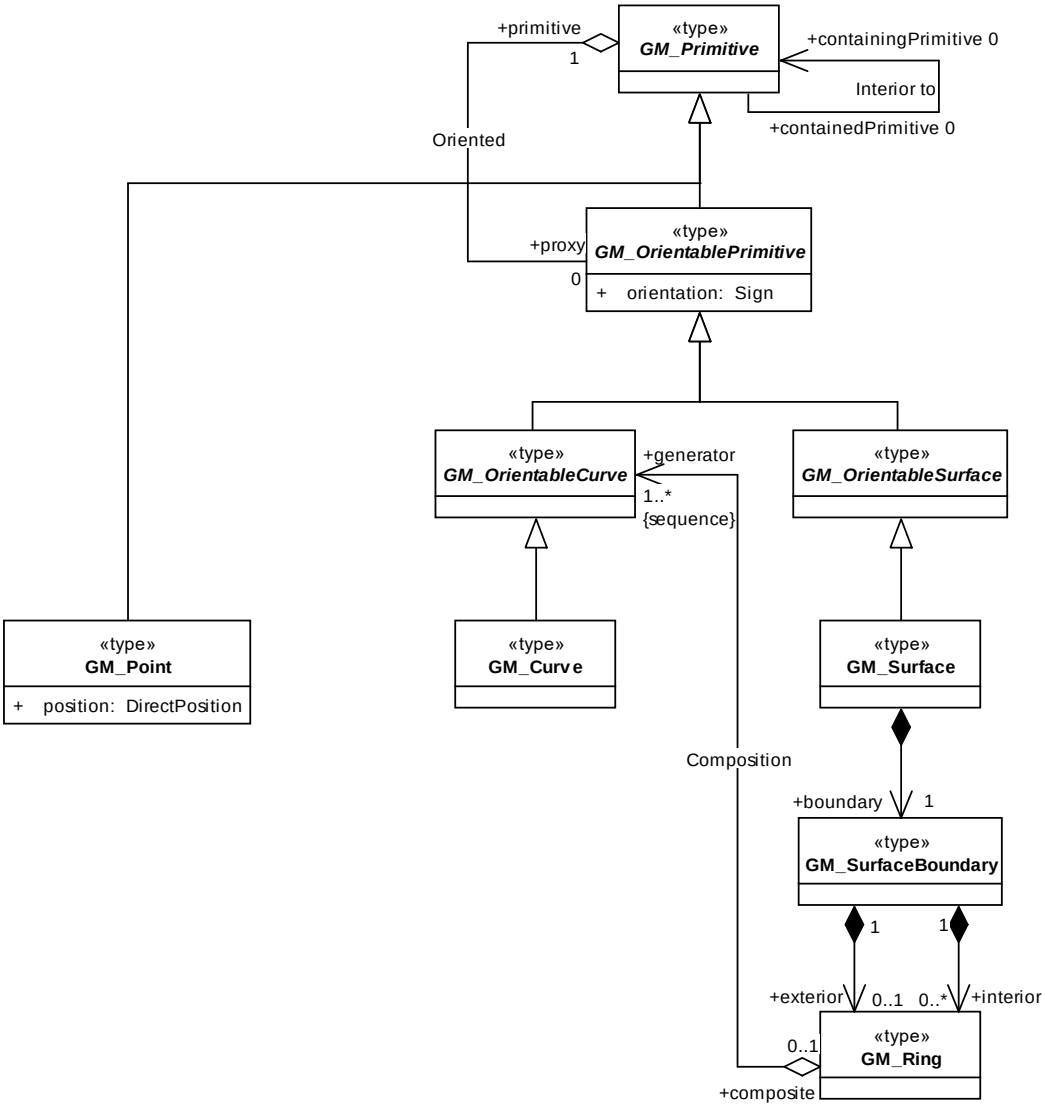


Figure 6.4.4 – Geometry primitive

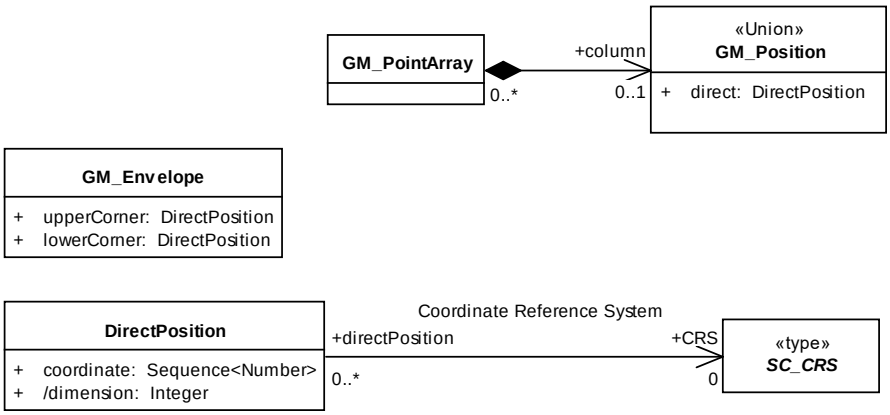


Figure 6.4.5 – Coordinate package

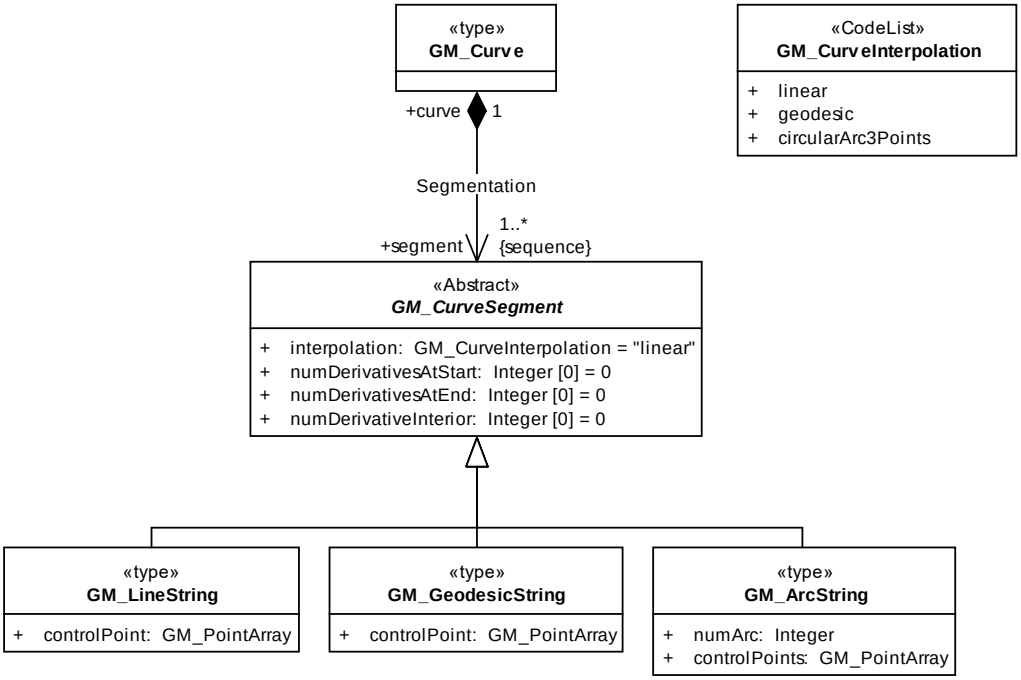


Figure 6.4.6 – Curve segment

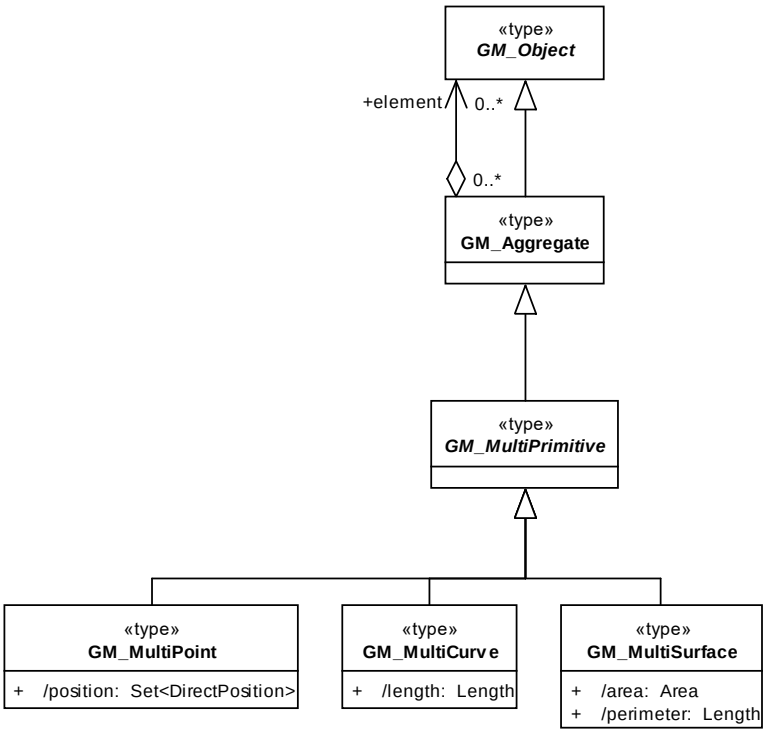


Figure 6.4.7 – Geometry aggregation

6.4.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.4.4.1(S1), 6.4.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.10 GM_Primitive – specializations 6.4.4.3(S4), 6.4.4.4(S6), 6.4.4.5(S8)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.4.4.6(S9)
- 6.3.15 GM_OrientableSurface – specializations 6.4.4.7(S10), 6.4.4.8(S12)
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specializations 6.4.4.8(S12), 6.4.4.10(S14)
- 6.4.1 DirectPosition – specialization 6.4.4.11(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.4.4.12(S18)
- 6.4.6 GM_PointArray
- 6.4.8 GM_CurveInterpolation – specialization 6.4.4.13(S19)
- 6.4.9 GM_CurveSegment – specializations 6.4.4.9(S13), 6.4.4.14(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString

- 6.4.14 GM_ArcString
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface

6.4.4 Specializations

6.4.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.4.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Object::envelope[0,1] : GM_Envelope
```

6.4.4.3 GM_Primitive (S4)

All GM_Primitives are simple.

```
context GM_Primitive inv:
self.isSimple() = TRUE
```

6.4.4.4 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.4.4.5 GM_Primitive (S8)

The Oriented association is not used. The multiplicity of the proxy role of the Oriented association between GM_Primitive and GM_OrientablePrimitive is constrained from [0,2] to [0].

```
GM_Primitive::proxy [0] : GM_OrientablePrimitive                       ([0,2])
```

6.4.4.6 GM_OrientableCurve (S9)

GM_OrientableCurve is abstract and not instantiable.

6.4.4.7 GM_OrientableSurface (S10)

GM_OrientableSurface is abstract and not instantiable.

6.4.4.8 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.4.4.9 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                   ([0,1])
```

6.4.4.10 GM_Surface (S14)

A GM_Surface is described by its boundary; GM_SurfacePatch is not used. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [0].

```
GM_Surface::patch[0] : GM_SurfacePatch                    ([0..n])
```

6.4.4.11 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS     ([0,1])
```

6.4.4.12 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                   ([0,1])
GM_Position::indirect[0] : GM_PointRef                    ([0,1])
```

6.4.4.13 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.4.4.14 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer       ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer      ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer         ([0,1])
```

6.5 3D Simple (L2.3D.3d – 3D no self intersection)

6.5.1 Introduction

This profile consists of a collection of 3D non-self intersecting geometric primitives. Simple geometry refers to data which has been captured with limited constraints. No connectivity is established between geometries but geometries are not allowed to self intersect. Thus, the primitives, as a collection, do not form a geometric complex, although subsets of the collection may coincidentally or expressly form complexes. Simple geometry can be useful if all that is required is a visual image or plot of a map and no spatial analysis is to be performed and is the first step in cleaning up spaghetti data.

6.5.2 Class diagrams

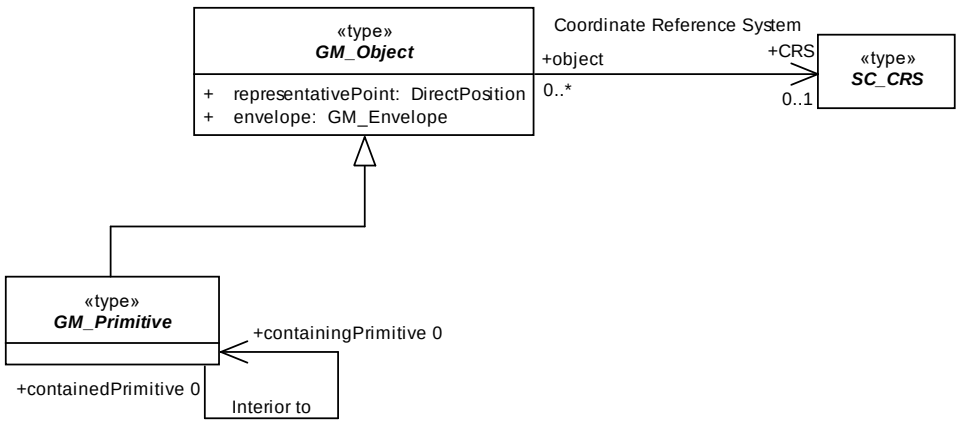


Figure 6.5.1 – Geometry object

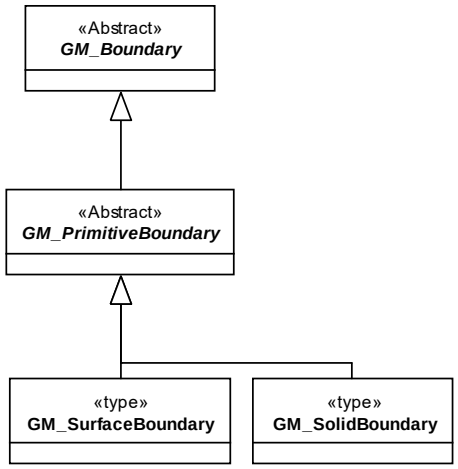


Figure 6.5.2 – Geometry boundary

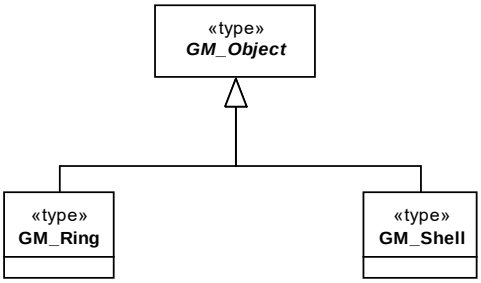


Figure 6.5.3 – Geometry boundary components

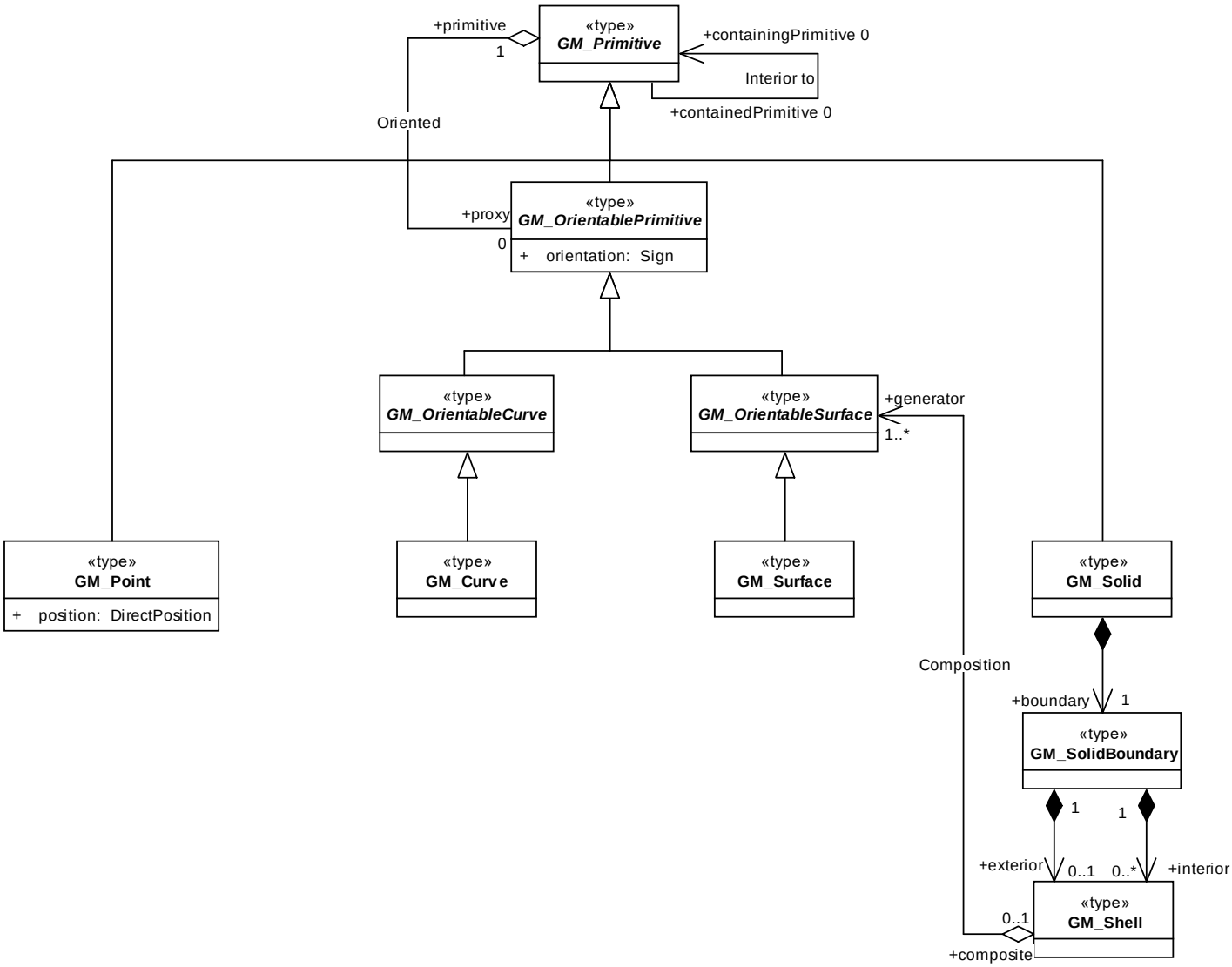


Figure 6.5.4 – Geometry primitive

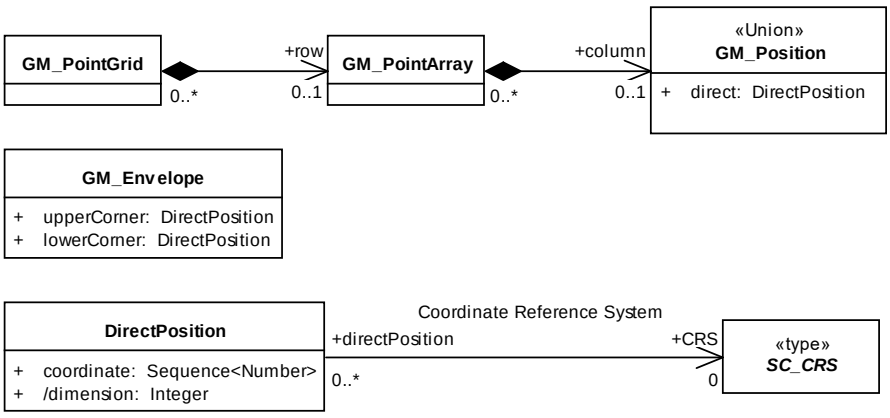


Figure 6.5.5 – Coordinate package

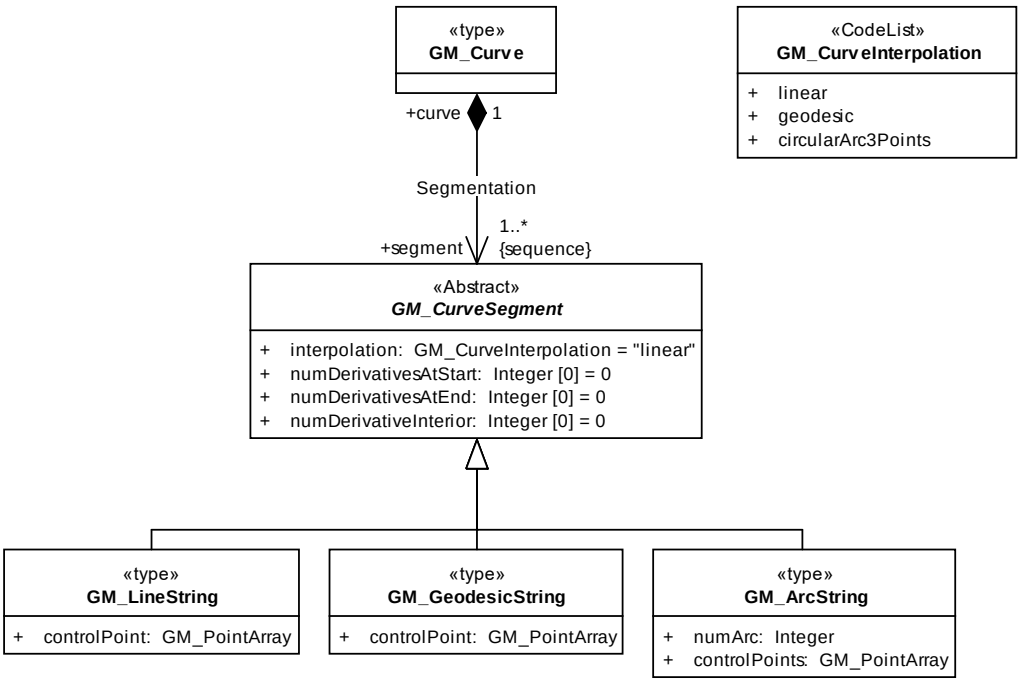


Figure 6.5.6 – Curve segment

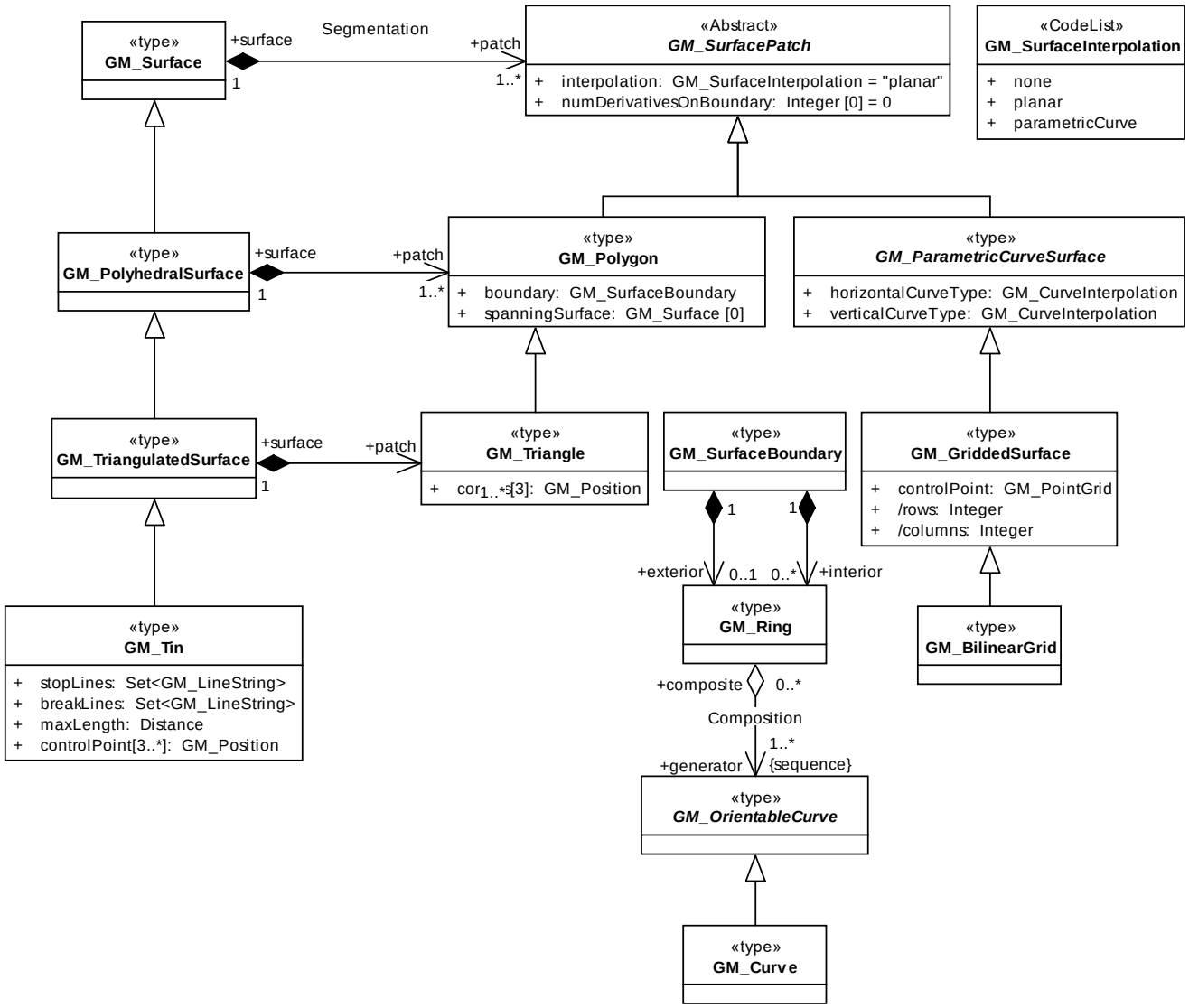


Figure 6.5.7 – Surface patch

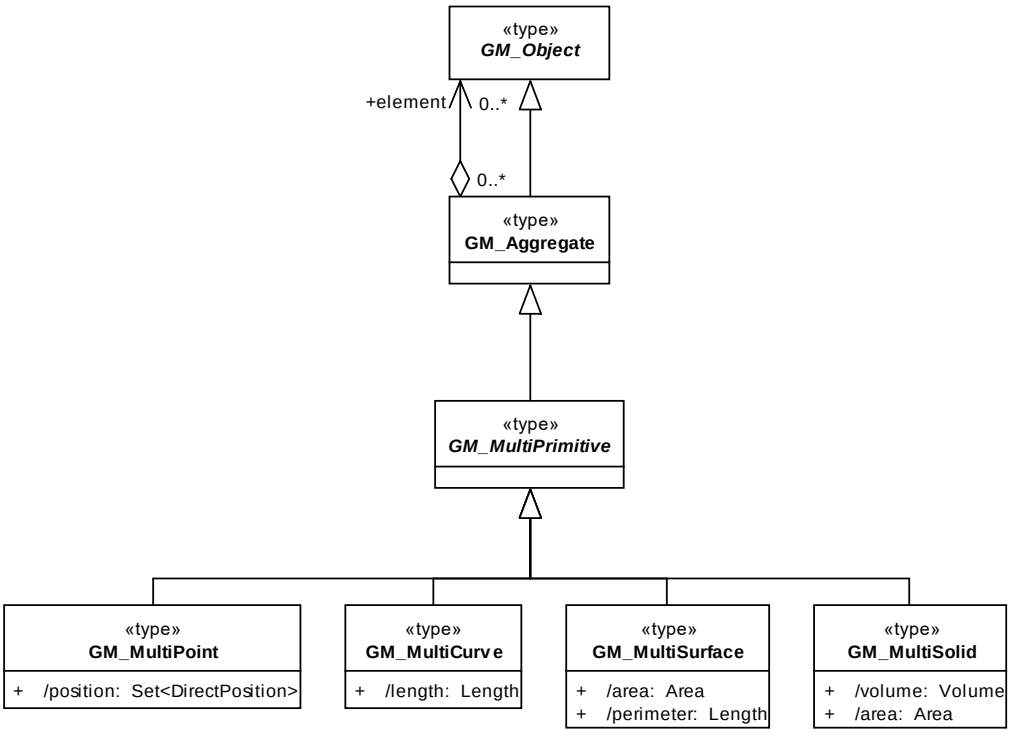


Figure 6.5.8 – Geometry aggregation

6.5.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.5.4.1(S1), 6.5.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring – specialization 6.5.4.3(S3)
- 6.3.7 GM_SurfaceBoundary – specialization 6.5.4.3(S3)
- 6.3.8 GM_Shell
- 6.3.9 GM_SolidBoundary
- 6.3.10 GM_Primitive – specializations 6.5.4.4(S4), 6.5.4.5(S6), 6.5.4.6(S8)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.5.4.7(S9)
- 6.3.15 GM_OrientableSurface – specialization 6.5.4.8(S10)
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specialization 6.5.4.10(S15)
- 6.3.18 GM_Solid – specialization 6.5.4.11(S16)
- 6.4.1 DirectPosition – specialization 6.5.4.12(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.5.4.13(S18)
- 6.4.6 GM_PointArray
- 6.4.6 GM_PointGrid

- 6.4.8 GM_CurveInterpolation – specialization 6.5.4.14(S19)
- 6.4.9 GM_CurveSegment – specializations 6.5.4.9(S13), 6.5.4.15(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.4.32 GM_SurfaceInterpolation – specialization 6.5.4.16(S21)
- 6.4.34 GM_SurfacePatch – specializations 6.5.4.17(S22), 6.5.4.18(S23)
- 6.4.35 GM_PolyhedralSurface
- 6.4.36 GM_Polygon – specialization 6.5.4.19(S24)
- 6.4.37 GM_TriangulatedSurface
- 6.4.38 GM_Triangle
- 6.4.39 GM_Tin
- 6.4.40 GM_ParametricCurveSurface
- 6.4.41 GM_GriddedSurface
- 6.4.45 GM_BilinearGrid
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.5.7 GM_MultiSolid

6.5.4 Specializations

6.5.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.5.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.5.4.3 GM_SurfaceBoundary, GM_Ring (S3)

GM_SurfaceBoundary and GM_Ring are only used to support GM_Polygon.

6.5.4.4 GM_Primitive (S4)

All GM_Primitives are simple.

```
context GM_Primitive inv:
self.isSimple() = TRUE
```

6.5.4.5 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.5.4.6 GM_Primitive (S8)

The Oriented association is not used. The multiplicity of the proxy role of the Oriented association between GM_Primitive and GM_OrientablePrimitive is constrained from [0,2] to [0].

```
GM_Primitive::proxy [0] : GM_OrientablePrimitive                       ([0,2])
```

6.5.4.7 GM_OrientableCurve (S9)

GM_OrientableCurve is abstract and not instantiable.

6.5.4.8 GM_OrientableSurface (S10)

GM_OrientableSurface is abstract and not instantiable.

6.5.4.9 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                    ([0,1])
```

6.5.4.10 GM_Surface (S15)

The interior of a GM_Surface patch is described by GM_SurfacePatches. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [1..n].

```
GM_Surface::patch[1..n] : GM_SurfacePatch                              ([0..n])
```

6.5.4.11 GM_Solid (S16)

The boundary operation of GM_Solid is changed to a composition relationship to GM_SolidBoundary.

```
GM_Solid::boundary[1] : GM_SolidBoundary
```

6.5.4.12 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                   ([0,1])
```

6.5.4.13 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                                 ([0,1])
GM_Position::indirect[0] : GM_PointRef                                  ([0,1])
```

6.5.4.14 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.5.4.15 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                     ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                       ([0,1])
```

6.5.4.16 GM_SurfaceInterpolation (S21)

The GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.

```
GM_SurfaceInterpolation::
none
planar
parametricCurve
```

6.5.4.17 GM_SurfacePatch (S22)

The multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained from [0,1] to [1].

```
GM_SurfacePatch::surface[1] : GM_Surface                    ([0,1])
```

6.5.4.18 GM_SurfacePatch (S23)

The multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained from [0,1] to [0].

```
GM_SurfacePatch::numDerivativesOnBoundary[0] : Integer      ([0,1])
```

6.5.4.19 GM_Polygon (S24)

The multiplicity of the spanningSurface attribute of GM_Polygon is constrained from [0..1] to [0].

```
GM_Polygon::spanningSurface[0] : GM_Surface                 ([0,1])
```

6.6 2D Shared (L3.2D.2d – 2D primitives disjoint)

6.6.1 Introduction

Primitives disjoint refers to data where primitives do not overlap, nor cross one another, nor are they duplicated. This can in some case be referred to as "clean" spaghetti data, which does not have explicit topological relationships but could be topologically structured quickly without needing a separate cleaning step to find and remove small gaps, overshoots, and slivers. 2D geographic primitives occur on a two-dimensional plane. Because the geometry exists only on a plane, lines that cross in the source data are broken into separate lines. In 2½D geospatial data collection when two lines cross at differing elevations in 3D they are treated as if they were in the 2D plane. The primitives are broken into separate lines at the intersection in X,Y but the separate Z values at the intersection may be retained. Thus the profile can handle multiple Z values at the same [X,Y] location (see H.2 for a discussion of Z-bust). In this profile primitives are bidirectional, whereas in Level 1 profiles primitives are unidirectional.

6.6.2 Class diagrams

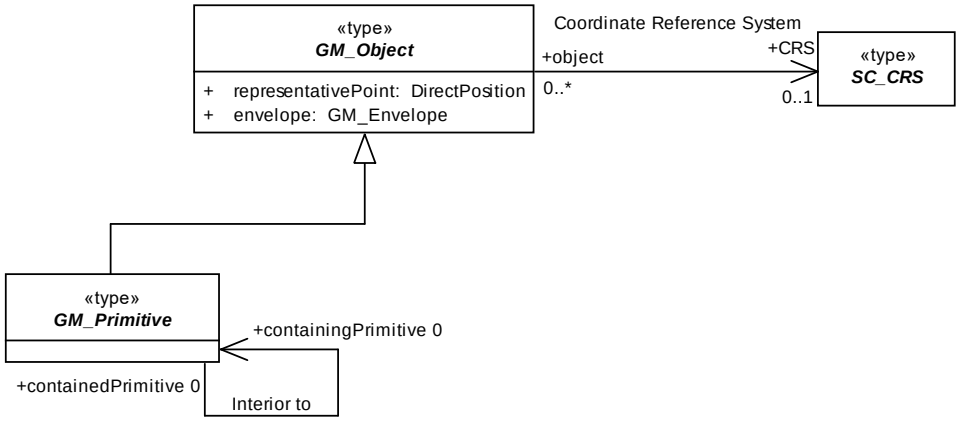


Figure 6.6.1 – Geometry object

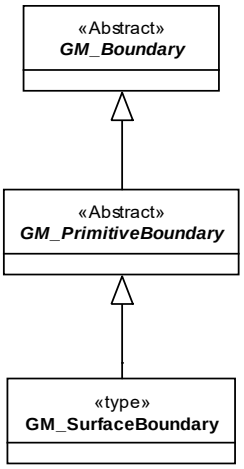


Figure 6.6.2 – Geometry boundary

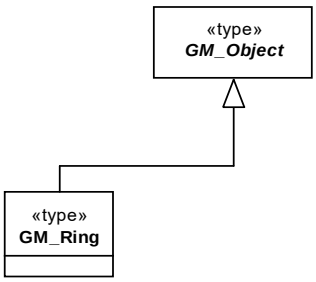


Figure 6.6.3 – Geometry boundary components

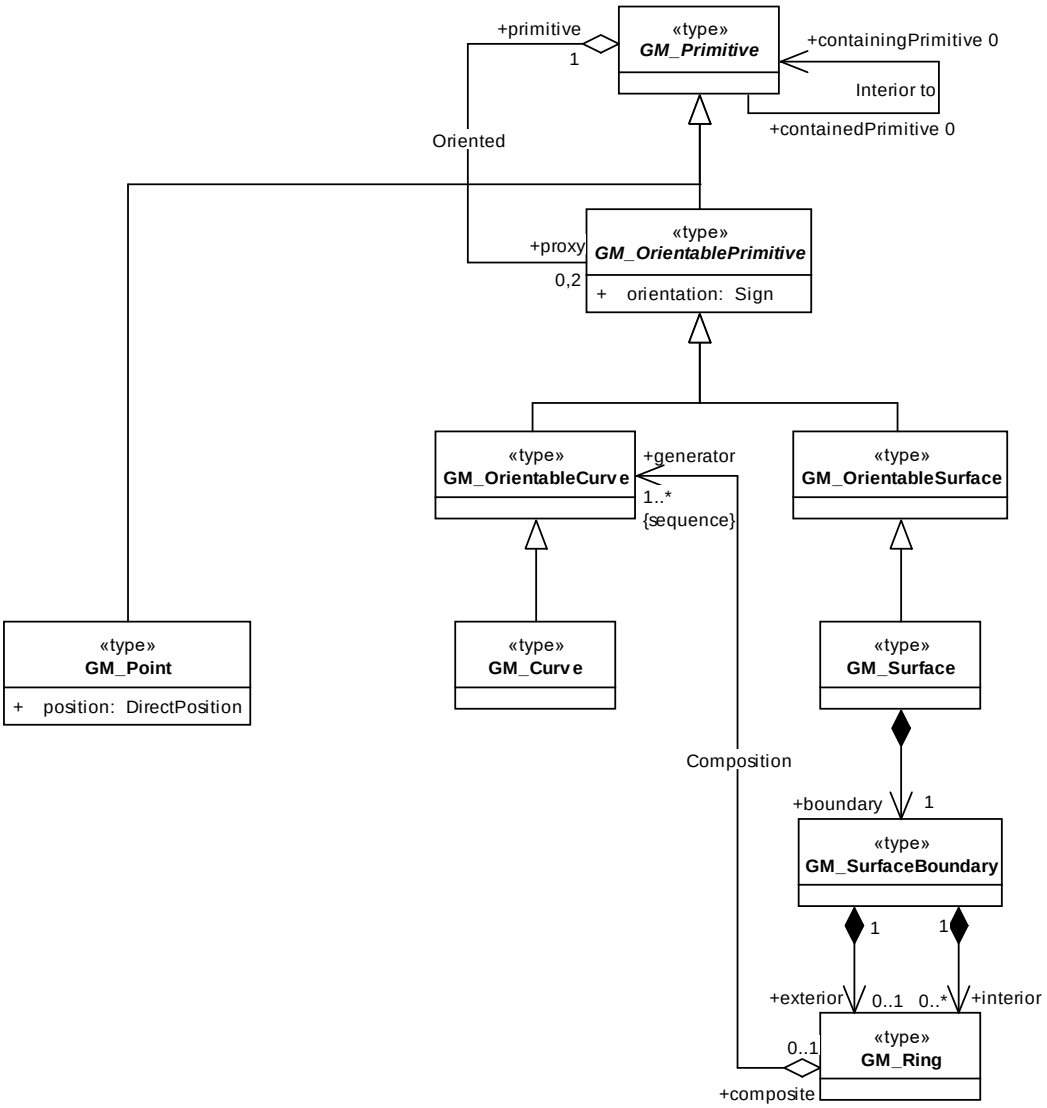


Figure 6.6.4 – Geometry primitive

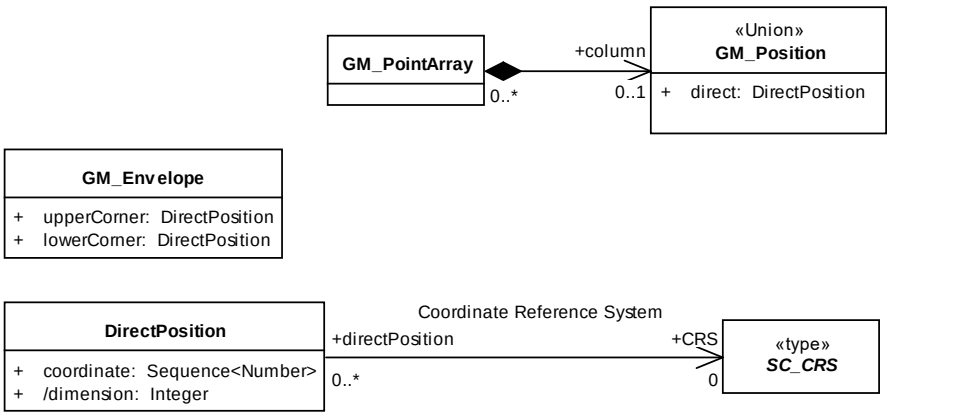


Figure 6.6.5 – Coordinate package

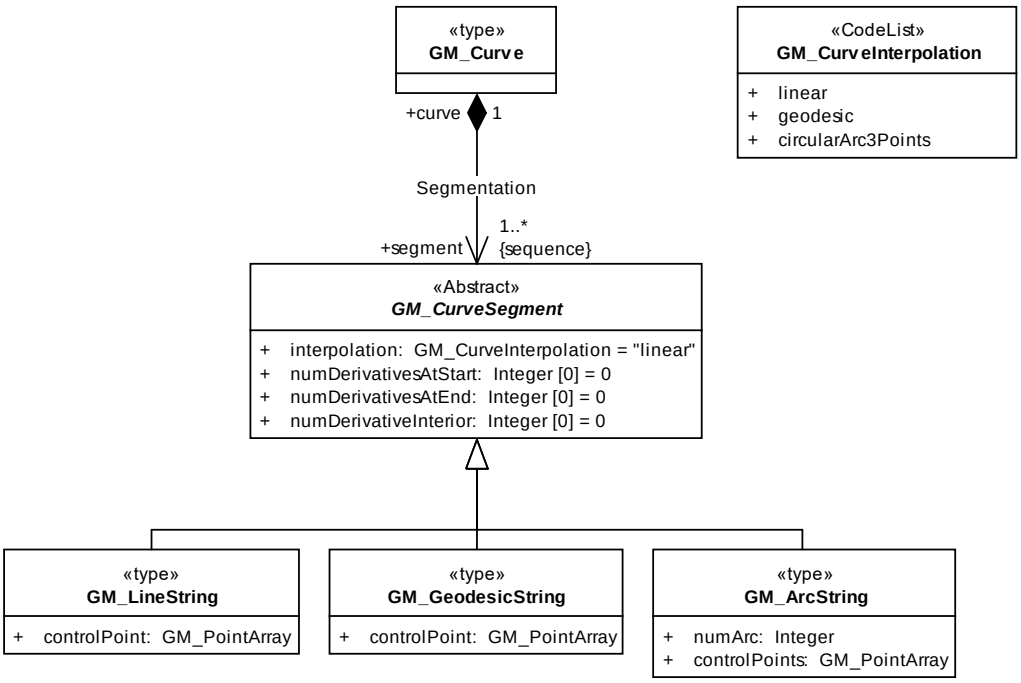


Figure 6.6.6 – Curve segment

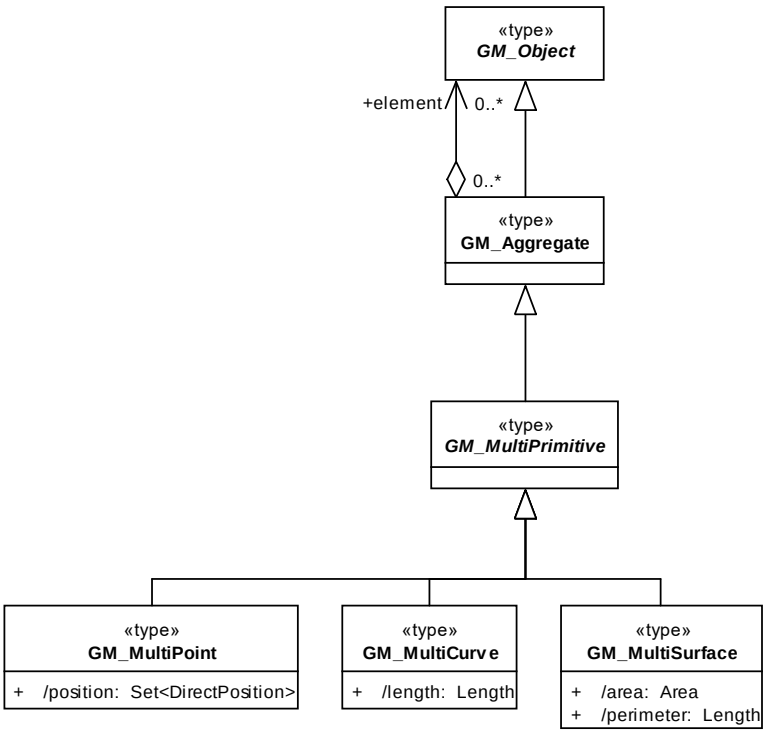


Figure 6.6.7 – Geometry aggregation

6.6.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.6.4.1(S1), 6.6.4.2(S2)

- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.10 GM_Primitive – specializations 6.6.4.3(S5), 6.6.4.4(S6)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve
- 6.3.15 GM_OrientableSurface – specialization 6.6.4.5(S12)
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specializations 6.6.4.5(S12), 6.6.4.7(S14)
- 6.4.1 DirectPosition – specialization 6.6.4.8(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.6.4.9(S18)
- 6.4.6 GM_PointArray
- 6.4.8 GM_CurveInterpolation – specialization 6.6.4.10(S19)
- 6.4.9 GM_CurveSegment – specializations 6.6.4.6(S13), 6.6.4.11(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface

6.6.4 Specializations

6.6.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.6.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.6.4.3 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.6.4.4 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.6.4.5 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.6.4.6 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                    ([0,1])
```

6.6.4.7 GM_Surface (S14)

A GM_Surface is described by its boundary; GM_SurfacePatch is not used. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [0].

```
GM_Surface::patch[0] : GM_SurfacePatch                                  ([0..n])
```

6.6.4.8 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                   ([0,1])
```

6.6.4.9 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                                 ([0,1])
GM_Position::indirect[0] : GM_PointRef                                  ([0,1])
```

6.6.4.10 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.6.4.11 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                     ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                       ([0,1])
```

6.7 3D Shared (L3.3D.3d – 3D primitives disjoint)

6.7.1 Introduction

Primitives disjoint refers to data where primitives do not overlap, nor cross one another, nor are they duplicated. This can in some case be referred to as "clean" spaghetti data, which does not have explicit topological relationships but could be topologically structured quickly without needing a separate cleaning step to find and remove small gaps, overshoots, and slivers. Lines that cross in the source data are broken into separate lines. 3D geographic primitives occur in a three-dimensional space. This introduces a “Z” or elevation component.

6.7.2 Class diagrams

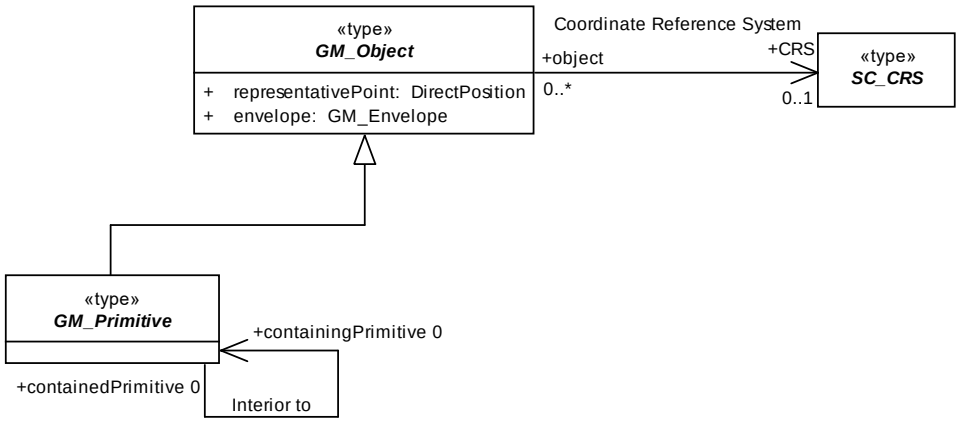


Figure 6.7.1 – Geometry object

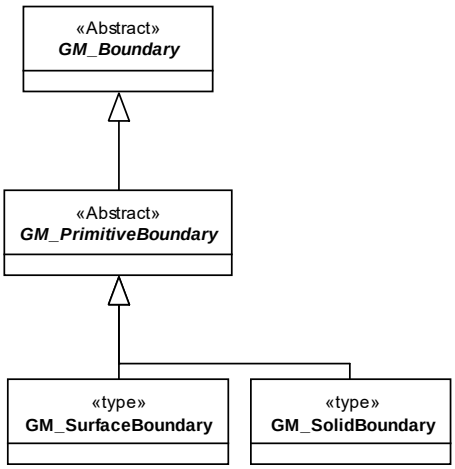


Figure 6.7.2 – Geometry boundary

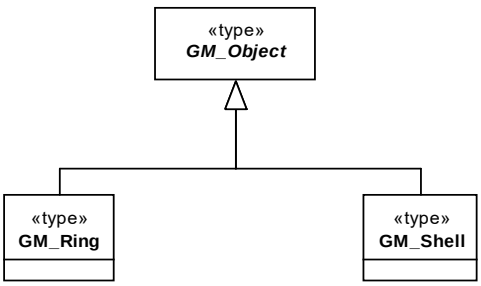


Figure 6.7.3 – Geometry boundary components

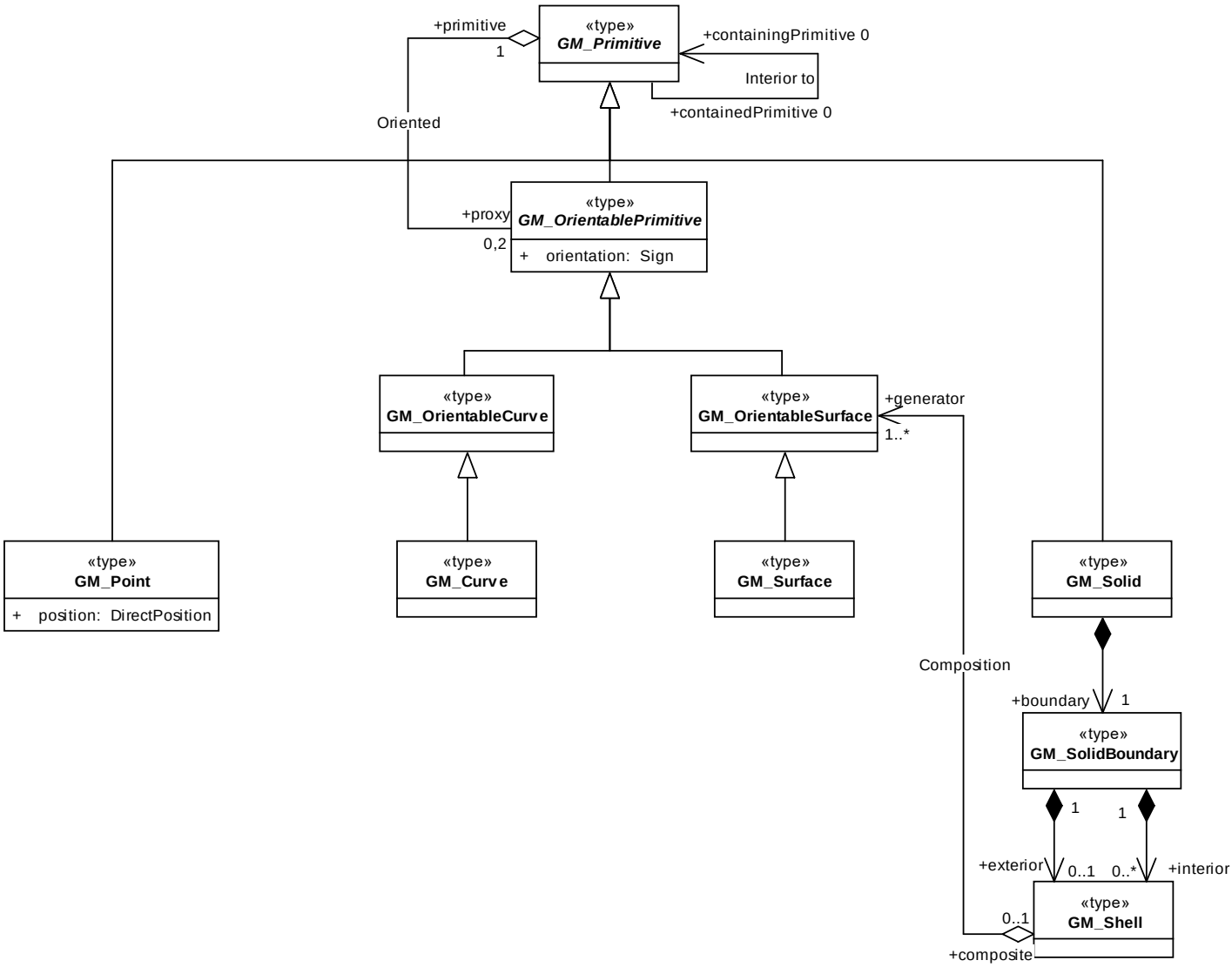


Figure 6.7.4 – Geometry primitive

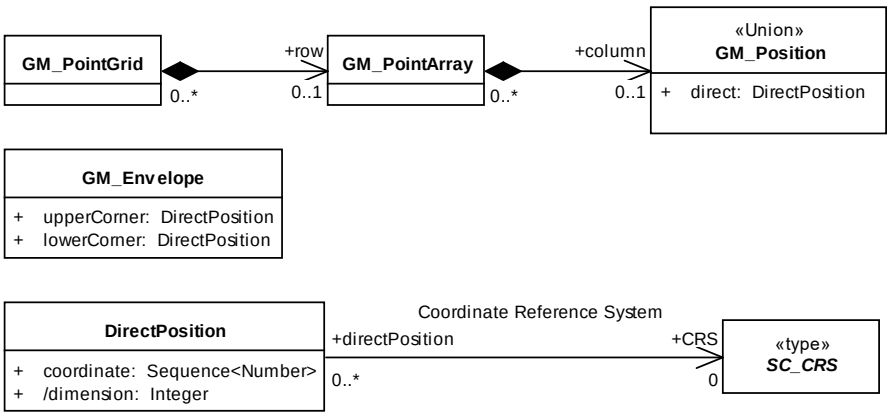


Figure 6.7.5 – Coordinate package

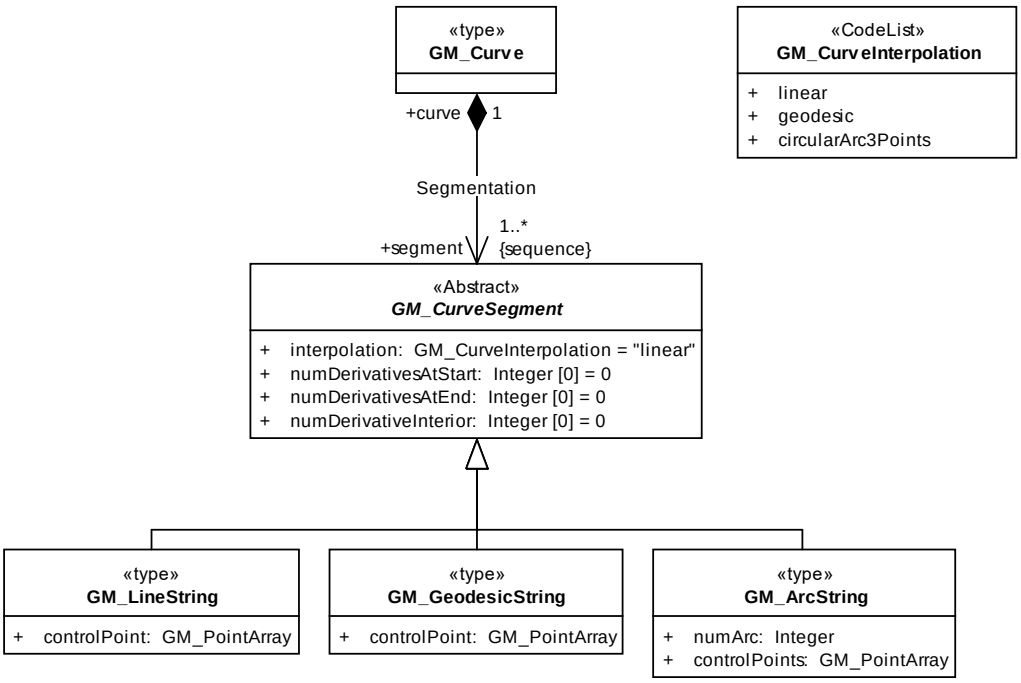


Figure 6.7.6 – Curve segment

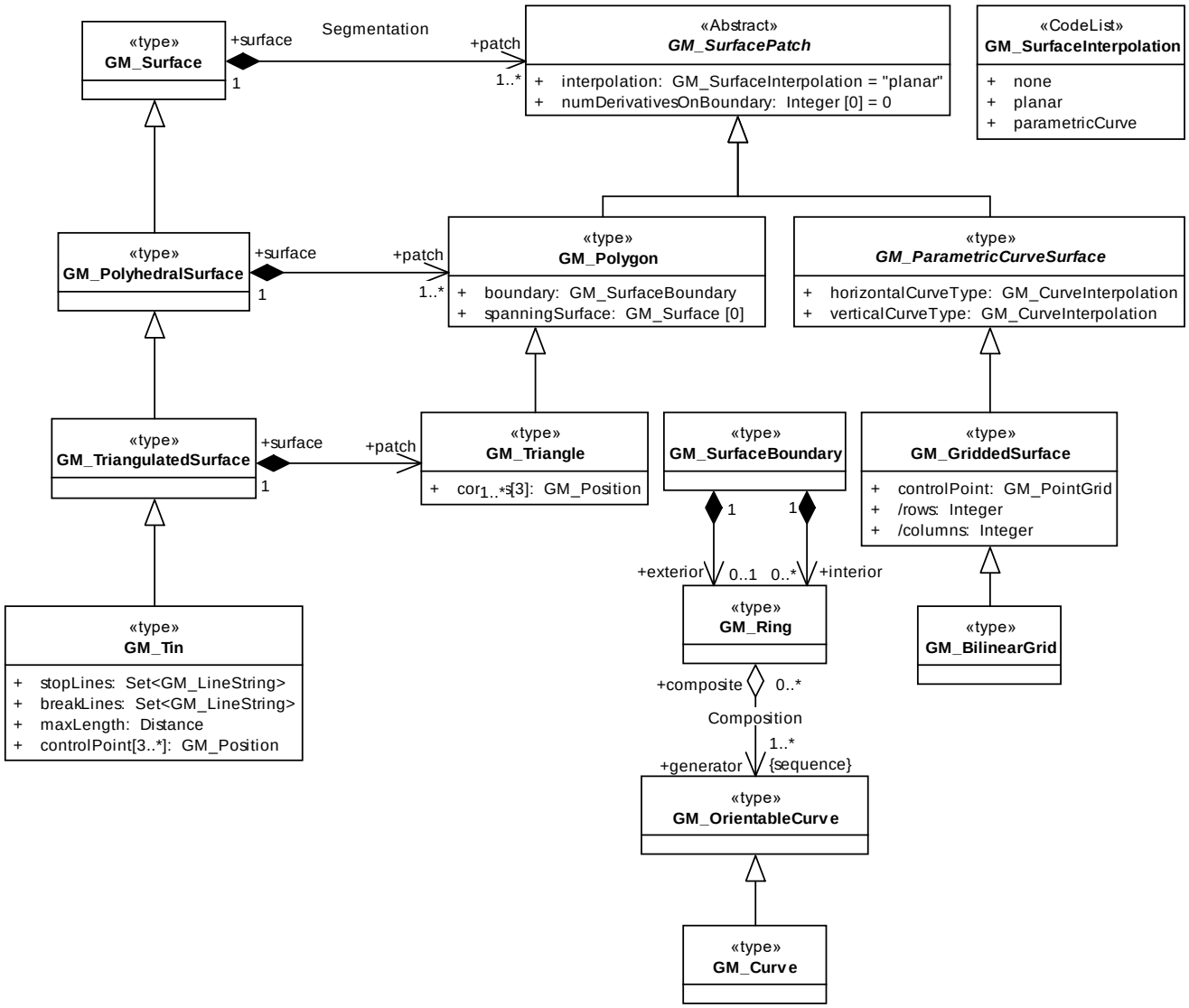


Figure 6.7.7 – Surface patch

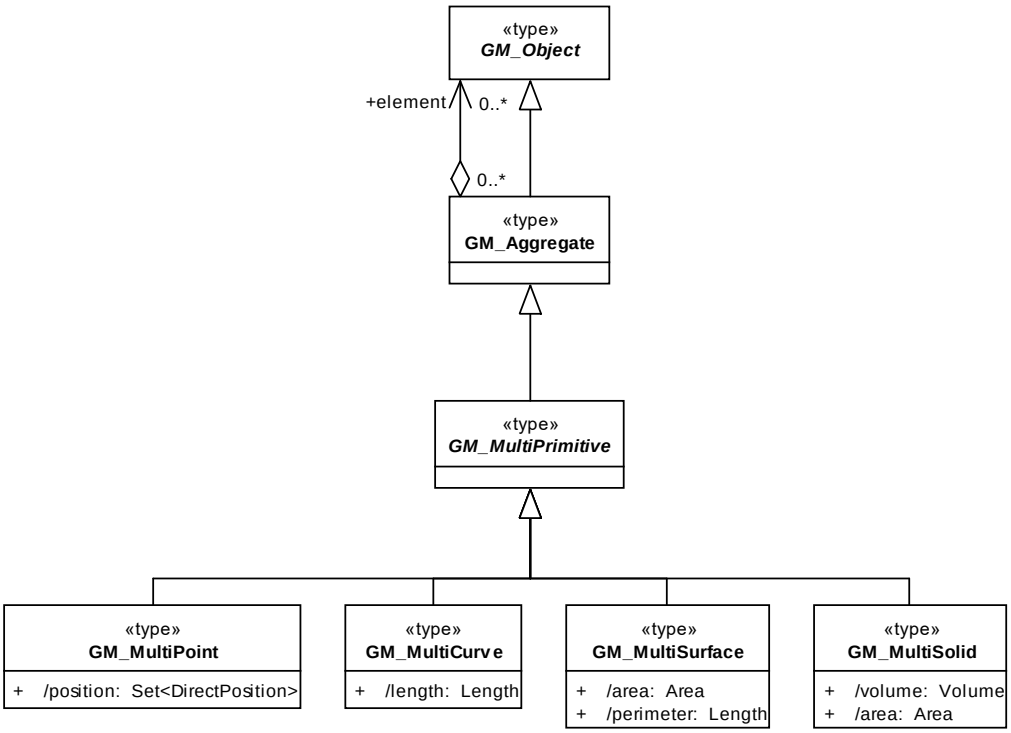


Figure 6.7.8 – Geometry aggregation

6.7.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.7.4.1(S1), 6.7.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring – specialization 6.7.4.3(S3)
- 6.3.7 GM_SurfaceBoundary – specializations 6.7.4.3(S3)
- 6.3.8 GM_Shell
- 6.3.9 GM_SolidBoundary
- 6.3.10 GM_Primitive – specializations 6.7.4.4(S5), 6.7.4.5(S6)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve
- 6.3.15 GM_OrientableSurface
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specialization 6.7.4.7(S15)
- 6.3.18 GM_Solid – specialization 6.7.4.8(S16)
- 6.4.1 DirectPosition – specialization 6.7.4.9(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.7.4.10(S18)
- 6.4.6 GM_PointArray
- 6.4.6 GM_PointGrid

- 6.4.8 GM_CurveInterpolation – specialization 6.7.4.11(S19)
- 6.4.9 GM_CurveSegment – specializations 6.7.4.6(S13), 6.7.4.12(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.4.32 GM_SurfaceInterpolation – specialization 6.7.4.13(S21)
- 6.4.34 GM_SurfacePatch – specializations 6.7.4.14(S22), 6.7.4.15(S23)
- 6.4.35 GM_PolyhedralSurface
- 6.4.36 GM_Polygon – specialization 6.7.4.16(S24)
- 6.4.37 GM_TriangulatedSurface
- 6.4.38 GM_Triangle
- 6.4.39 GM_Tin
- 6.4.40 GM_ParametricCurveSurface
- 6.4.41 GM_GriddedSurface
- 6.4.45 GM_BilinearGrid
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.5.7 GM_MultiSolid

6.7.4 Specializations

6.7.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.7.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.7.4.3 GM_SurfaceBoundary, GM_Ring (S3)

GM_SurfaceBoundary and GM_Ring are only used to support GM_Polygon.

6.7.4.4 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.7.4.5 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.7.4.6 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                   ([0,1])
```

6.7.4.7 GM_Surface (S15)

The interior of a GM_Surface patch is described by GM_SurfacePatches. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [1..n].

```
GM_Surface::patch[1..n] : GM_SurfacePatch                    ([0..n])
```

6.7.4.8 GM_Solid (S16)

The boundary operation of GM_Solid is changed to a composition relationship to GM_SolidBoundary.

```
GM_Solid::boundary[1] : GM_SolidBoundary
```

6.7.4.9 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS        ([0,1])
```

6.7.4.10 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                      ([0,1])
GM_Position::indirect[0] : GM_PointRef                       ([0,1])
```

6.7.4.11 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.7.4.12 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer          ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer         ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer            ([0,1])
```

6.7.4.13 GM_SurfaceInterpolation (S21)

The GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.

```
GM_SurfaceInterpolation::
none
planar
parametricCurve
```

6.7.4.14 GM_SurfacePatch (S22)

The multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained from [0,1] to [1].

```
GM_SurfacePatch::surface[1] : GM_Surface                     ([0,1])
```

6.7.4.15 GM_SurfacePatch (S23)

The multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained from [0,1] to [0].

```
GM_SurfacePatch::numDerivativesOnBoundary[0] : Integer       ([0,1])
```

6.7.4.16 GM_Polygon (S24)

The multiplicity of the spanningSurface attribute of GM_Polygon is constrained from [0..1] to [0].

`GM_Polygon::spanningSurface[0] : GM_Surface ([0,1])`

6.8 2D Complex (L4.2D.2d – 2D complex)

6.8.1 Introduction

This profile consists of a collection of 2D geometric primitives that form a complex. A complex refers to spatial data where primitive interiors are disjoint and all boundary primitives exist. Having all boundary primitives is a prerequisite to establishing topological relationships. This would be a useful profile in a Service Oriented Architecture (SOA) where a Web Service builds topology on the fly.

6.8.2 Class diagrams

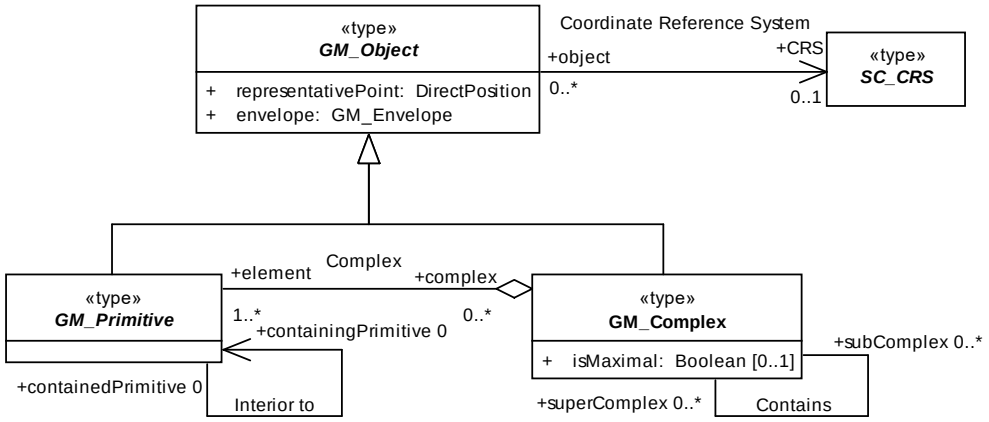


Figure 6.8.1 – Geometry object

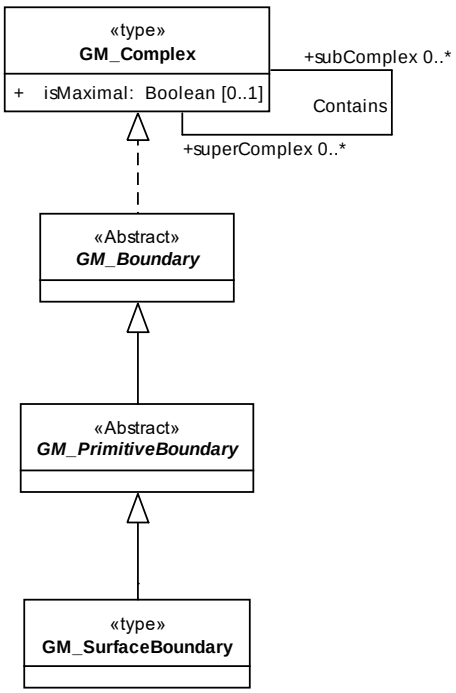


Figure 6.8.2 – Geometry boundary

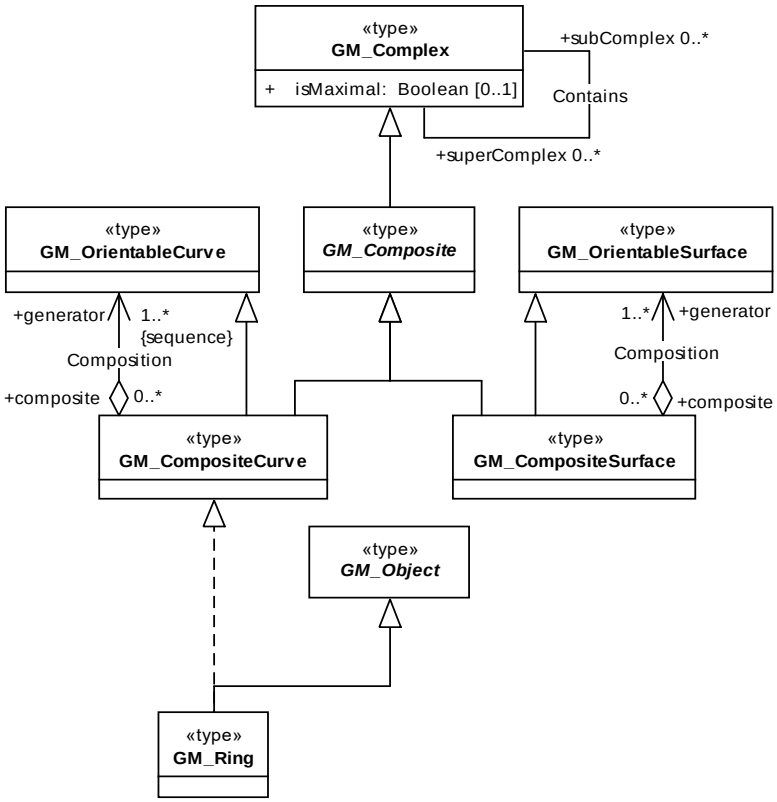


Figure 6.8.3 – Geometry boundary components

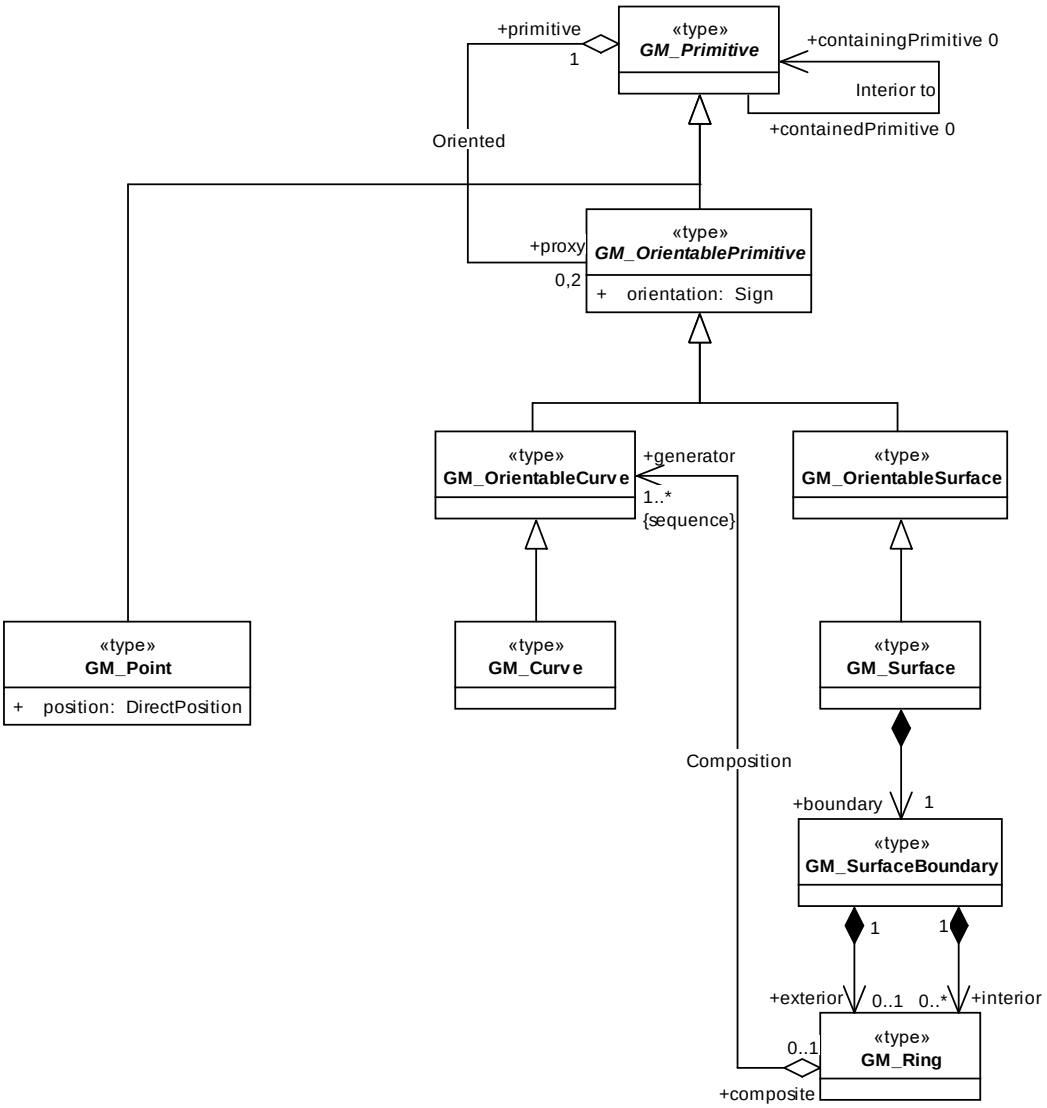


Figure 6.8.4 – Geometry primitive

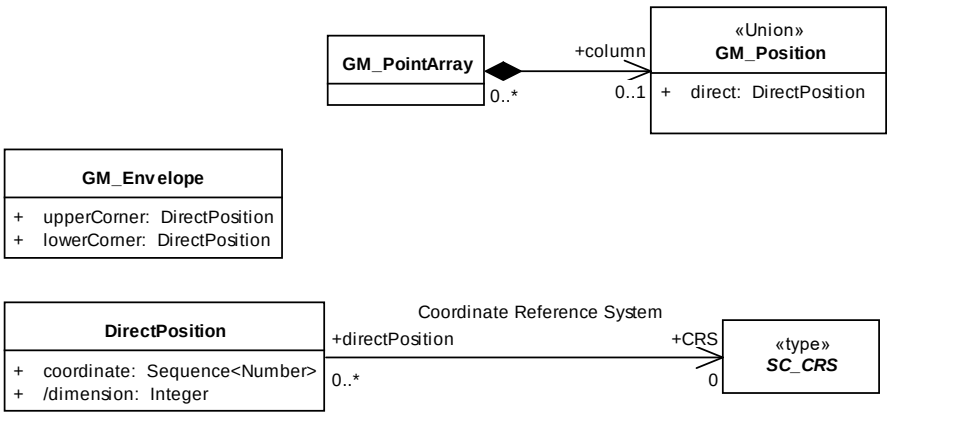


Figure 6.8.5 – Coordinate package

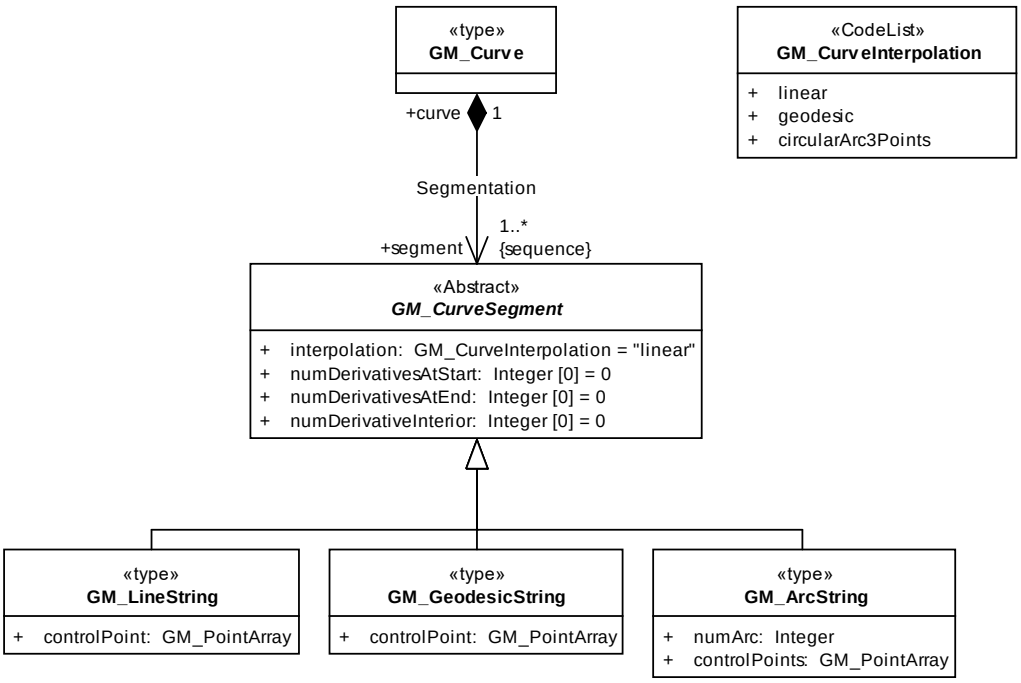


Figure 6.8.6 – Curve segment

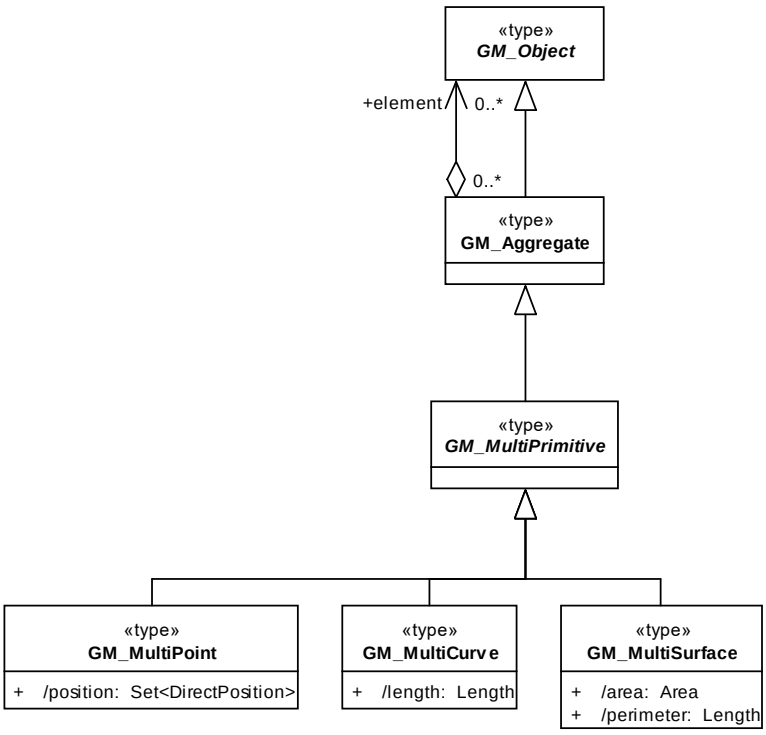


Figure 6.8.7 – Geometry aggregation

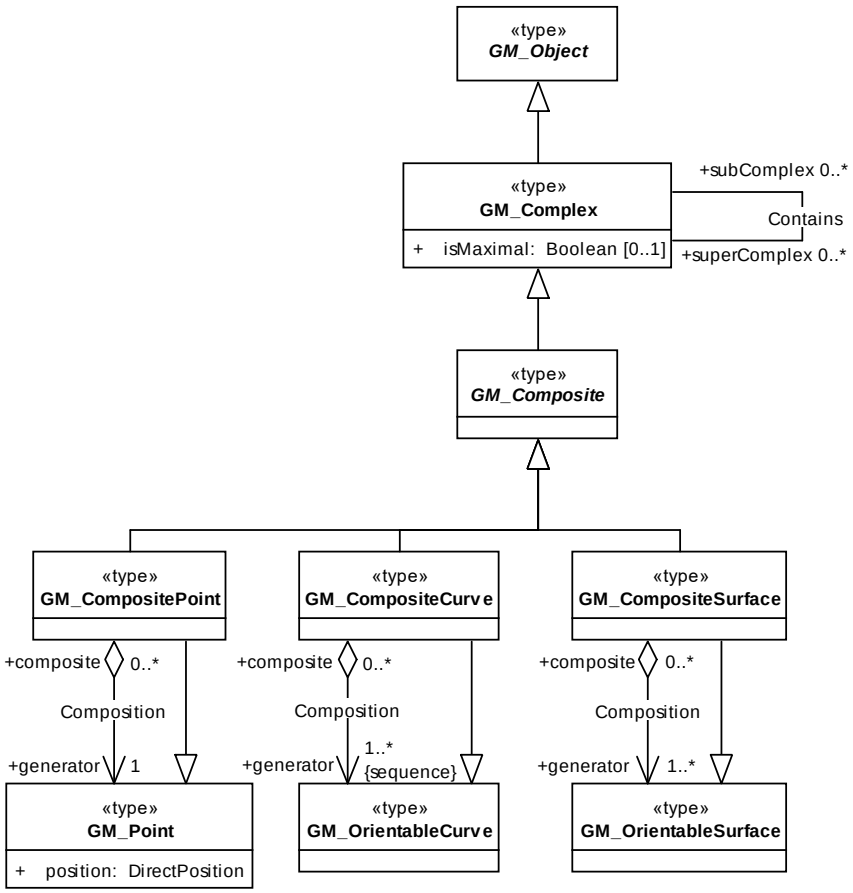


Figure 6.8.8 – Geometry composites

6.8.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.8.4.1(S1), 6.8.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.10 GM_Primitive – specializations 6.8.4.3(S5), 6.8.4.4(S6)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve
- 6.3.15 GM_OrientableSurface – specialization 6.8.4.5(S12)
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specializations 6.8.4.5(S12), 6.8.4.7(S14)
- 6.4.1 DirectPosition – specialization 6.8.4.8(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.8.4.9(S18)
- 6.4.6 GM_PointArray

- 6.4.8 GM_CurveInterpolation – specialization 6.8.4.10(S19)
- 6.4.9 GM_CurveSegment – specializations 6.8.4.6(S13), 6.8.4.11(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.6.2 GM_Complex – specialization 6.8.4.12(S25)
- 6.6.3 GM_Composite
- 6.6.4 GM_CompositePoint
- 6.6.5 GM_CompositeCurve
- 6.6.6 GM_CompositeSurface

6.8.4 Specializations

6.8.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.8.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.8.4.3 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.8.4.4 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.8.4.5 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.8.4.6 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                  ([0,1])
```

6.8.4.7 GM_Surface (S14)

A GM_Surface is described by its boundary; GM_SurfacePatch is not used. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [0].

```
GM_Surface::patch[0] : GM_SurfacePatch                                  ([0..n])
```

6.8.4.8 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                   ([0,1])
```

6.8.4.9 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                                 ([0,1])
GM_Position::indirect[0] : GM_PointRef                                  ([0,1])
```

6.8.4.10 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.8.4.11 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                     ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                       ([0,1])
```

6.8.4.12 GM_Complex (S25)

The isMaximal operation of GM_Complex is changed to an optional Boolean attribute.

```
GM_Complex::isMaximal[0,1] : Boolean
```

6.9 3D Complex (L4.3D.3d – 3D complex)

6.9.1 Introduction

This profile consists of a collection of 3D geometric primitives that form a complex. A complex refers to spatial data where primitive interiors are disjoint and all boundary primitives exist. Having all boundary primitives is a prerequisite to establishing topological relationships. This would be a useful profile in a Service Oriented Architecture (SOA) where a Web Service builds topology on the fly.

6.9.2 Class diagrams

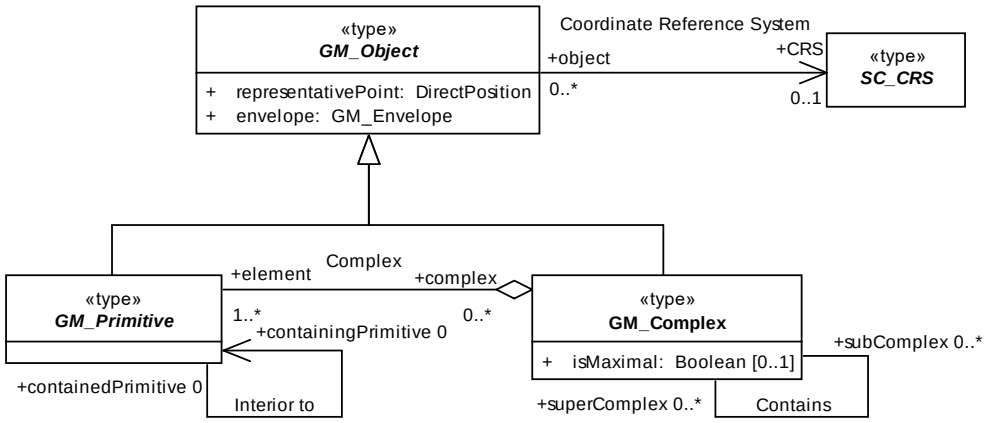


Figure 6.9.1 – Geometry object

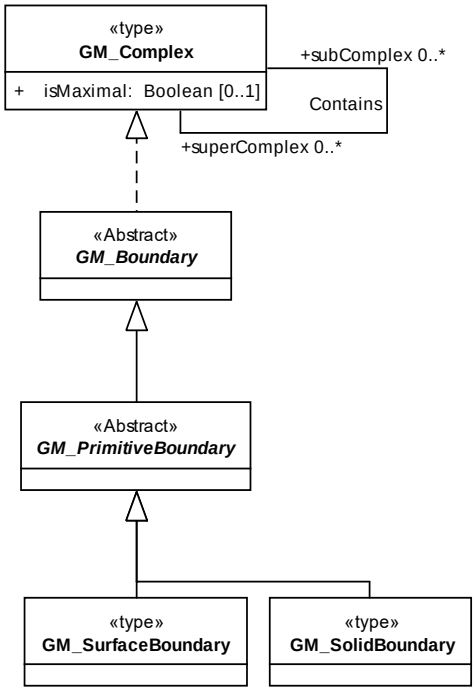


Figure 6.9.2 – Geometry boundary

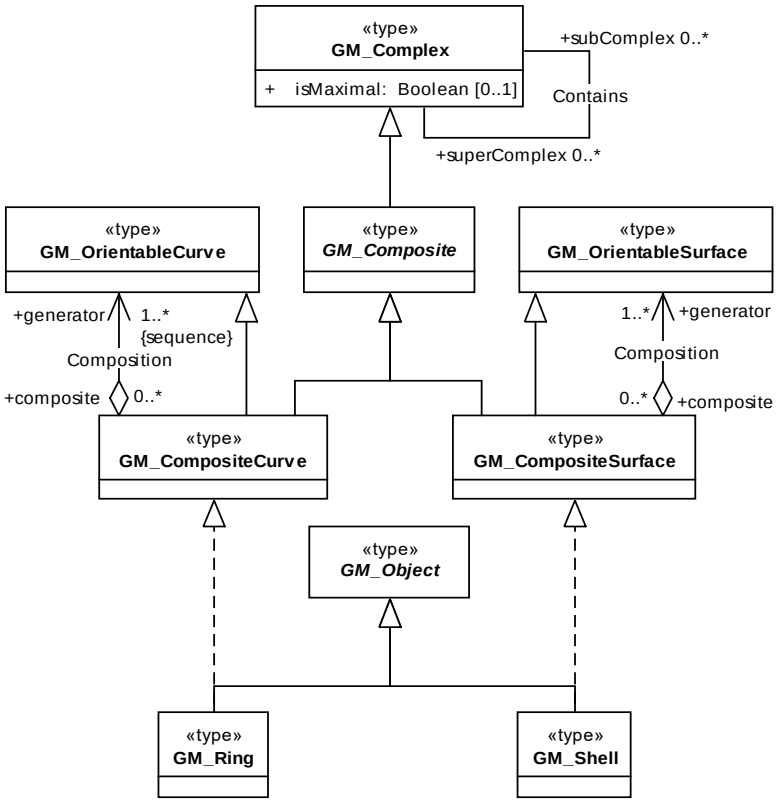


Figure 6.9.3 – Geometry boundary components

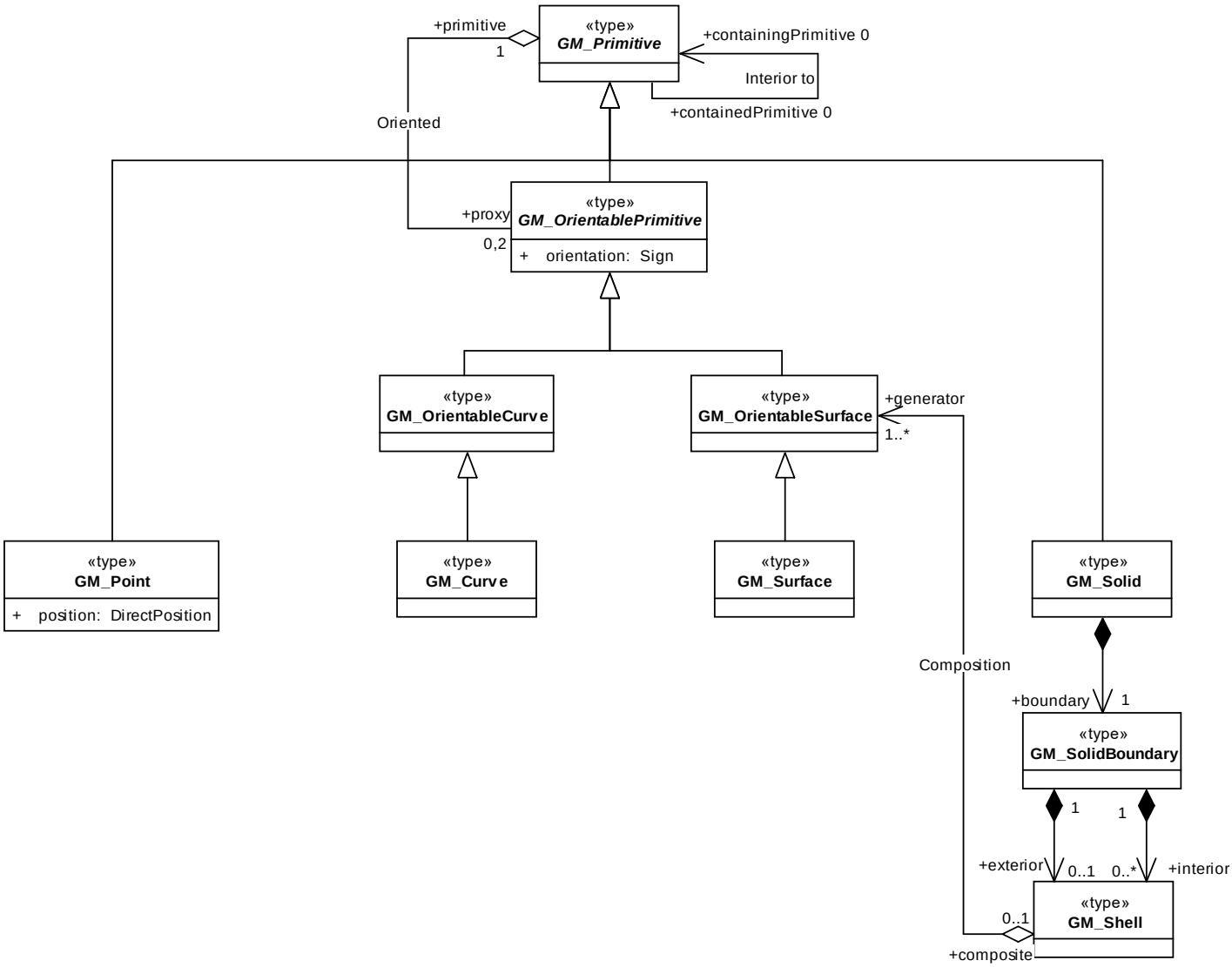


Figure 6.9.4 – Geometry primitive

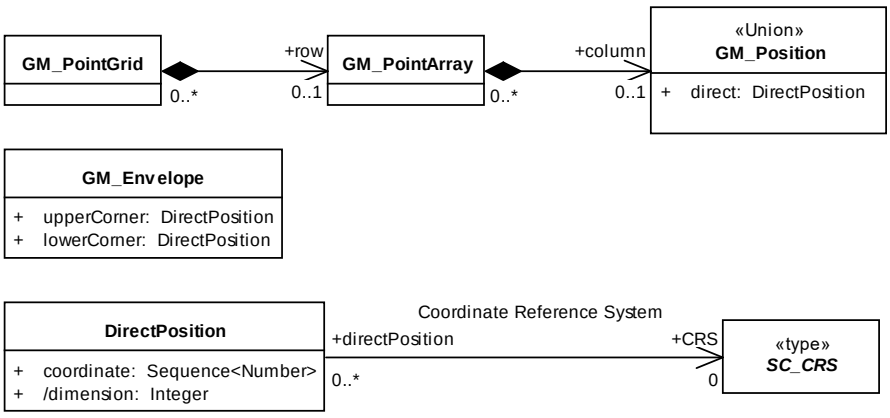


Figure 6.9.5 – Coordinate package

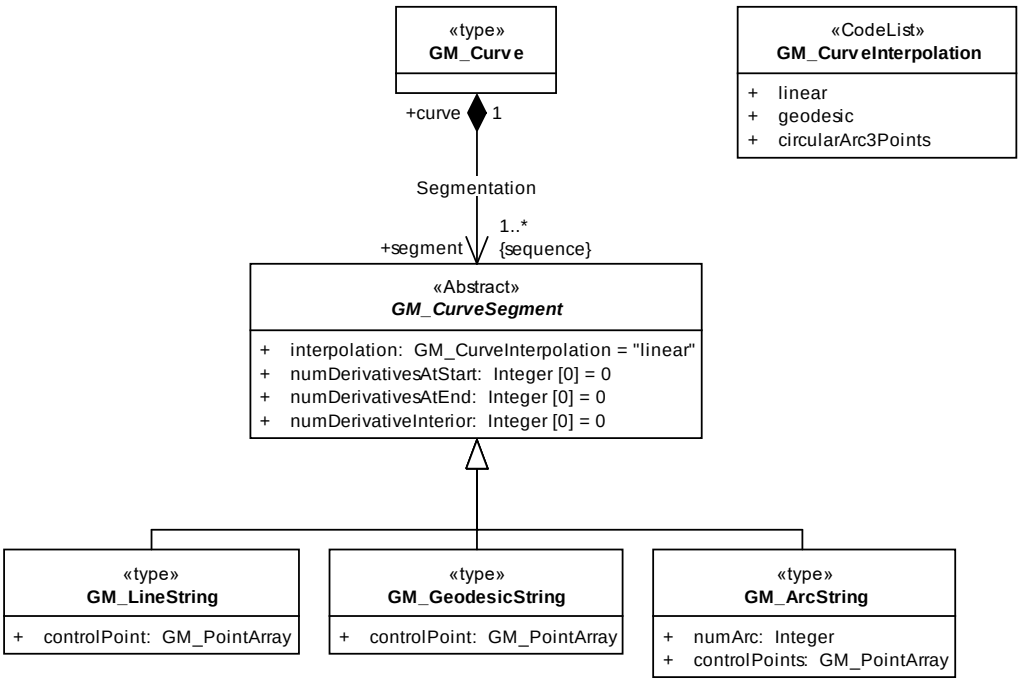


Figure 6.9.6 – Curve segment

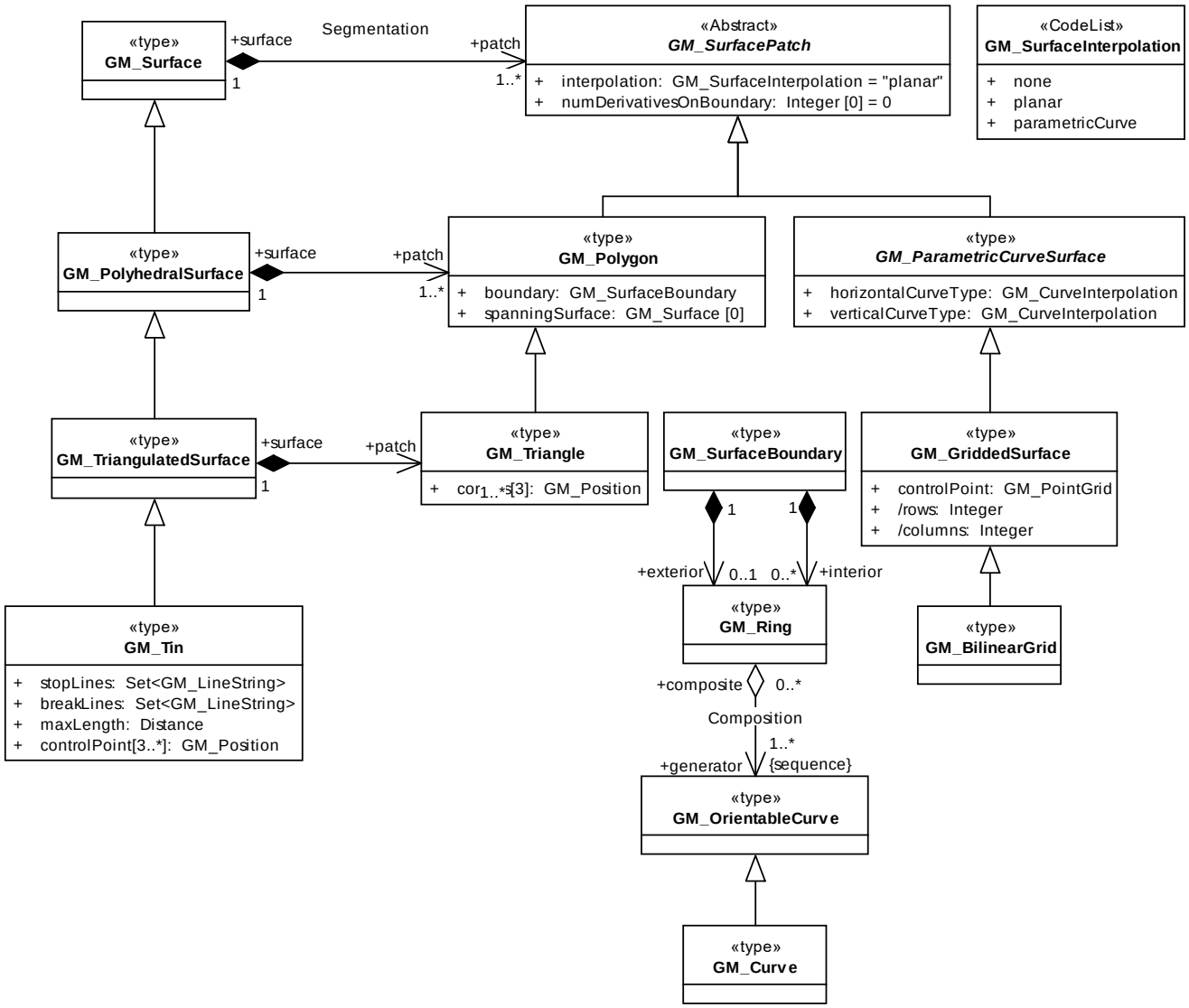


Figure 6.9.7 – Surface patch

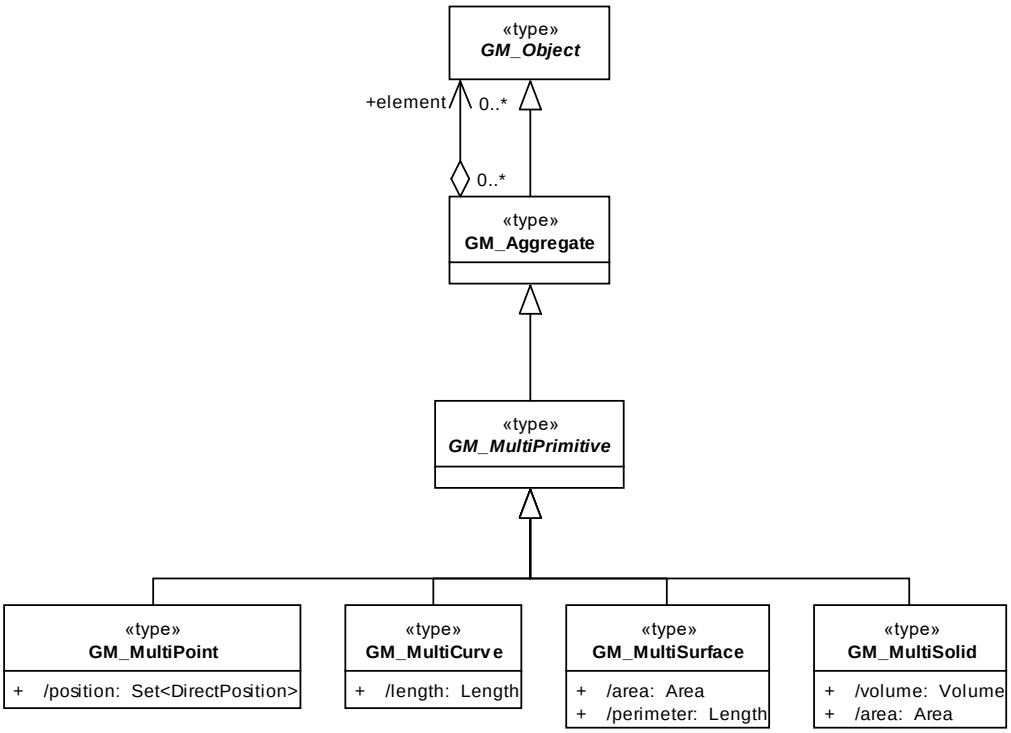


Figure 6.9.8 – Geometry aggregation

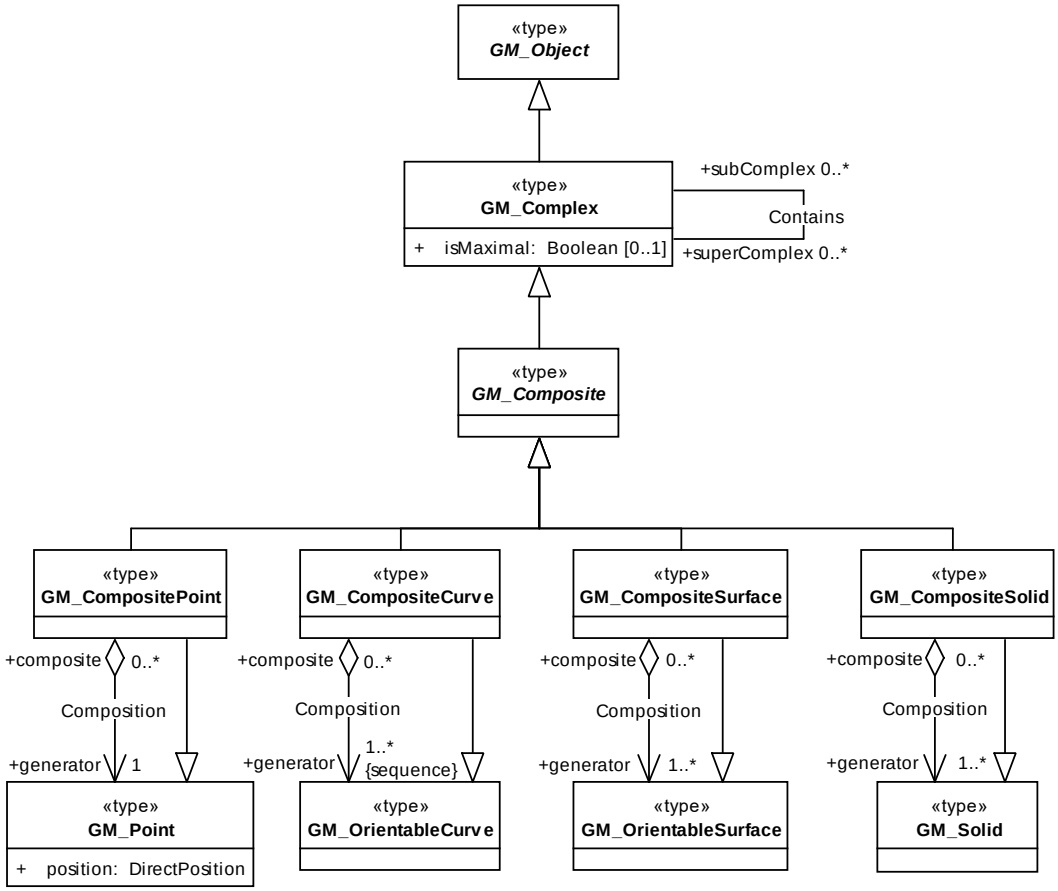


Figure 6.9.9 – Geometry composites

6.9.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.9.4.1(S1), 6.9.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.6 GM_Ring – specialization 6.9.4.3(S3)
- 6.3.7 GM_SurfaceBoundary – specialization 6.9.4.3(S3)
- 6.3.8 GM_Shell
- 6.3.9 GM_SolidBoundary
- 6.3.10 GM_Primitive – specializations 6.9.4.4(S5), 6.9.4.5(S6)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve
- 6.3.15 GM_OrientableSurface
- 6.3.16 GM_Curve
- 6.3.17 GM_Surface – specialization 6.9.4.7(S15)
- 6.3.18 GM_Solid – specialization 6.9.4.8(S16)
- 6.4.1 DirectPosition – specialization 6.9.4.9(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.9.4.10(S18)
- 6.4.6 GM_PointArray
- 6.4.6 GM_PointGrid
- 6.4.8 GM_CurveInterpolation – specialization 6.9.4.11(S19)
- 6.4.9 GM_CurveSegment – specializations 6.9.4.6(S13), 6.9.4.12(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.4.32 GM_SurfaceInterpolation – specialization 6.9.4.13(S21)
- 6.4.34 GM_SurfacePatch – specializations 6.9.4.14(S22), 6.9.4.15(S23)
- 6.4.35 GM_PolyhedralSurface
- 6.4.36 GM_Polygon – specialization 6.9.4.16(S24)
- 6.4.37 GM_TriangulatedSurface
- 6.4.38 GM_Triangle
- 6.4.39 GM_Tin
- 6.4.40 GM_ParametricCurveSurface
- 6.4.41 GM_GriddedSurface
- 6.4.45 GM_BilinearGrid
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.5.7 GM_MultiSolid

- 6.6.2 GM_Complex – specialization 6.9.4.17(S25)
- 6.6.3 GM_Composite
- 6.6.4 GM_CompositePoint
- 6.6.5 GM_CompositeCurve
- 6.6.6 GM_CompositeSurface
- 6.6.7 GM_CompositeSolid

6.9.4 Specializations

6.9.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.9.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.9.4.3 GM_SurfaceBoundary, GM_Ring (S3)

GM_SurfaceBoundary and GM_Ring are only used to support GM_Polygon.

6.9.4.4 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.9.4.5 GM_Primitive (S6)

The InteriorTo association of GM_Primitive is not used. The multiplicity of the containedPrimitive and containingPrimitive association roles are constrained from [0..n] to [0].

```
GM_Primitive::containedPrimitive[0] : GM_Primitive                    ([0..n])
GM_Primitive::containingPrimitive[0] : GM_Primitive                   ([0..n])
```

6.9.4.6 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                   ([0,1])
```

6.9.4.7 GM_Surface (S15)

The interior of a GM_Surface patch is described by GM_SurfacePatches. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [1..n].

```
GM_Surface::patch[1..n] : GM_SurfacePatch                              ([0..n])
```

6.9.4.8 GM_Solid (S16)

The boundary operation of GM_Solid is changed to a composition relationship to GM_SolidBoundary.

```
GM_Solid::boundary[1] : GM_SolidBoundary
```

6.9.4.9 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                  ([0,1])
```

6.9.4.10 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                    ([0,1])
GM_Position::indirect[0] : GM_PointRef                     ([0,1])
```

6.9.4.11 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.9.4.12 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer        ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer       ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer          ([0,1])
```

6.9.4.13 GM_SurfaceInterpolation (S21)

The GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.

```
GM_SurfaceInterpolation::
none
planar
parametricCurve
```

6.9.4.14 GM_SurfacePatch (S22)

The multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained from [0,1] to [1].

```
GM_SurfacePatch::surface[1] : GM_Surface                   ([0,1])
```

6.9.4.15 GM_SurfacePatch (S23)

The multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained from [0,1] to [0].

```
GM_SurfacePatch::numDerivativesOnBoundary[0] : Integer     ([0,1])
```

6.9.4.16 GM_Polygon (S24)

The multiplicity of the spanningSurface attribute of GM_Polygon is constrained from [0..1] to [0].

```
GM_Polygon::spanningSurface[0] : GM_Surface                ([0,1])
```

6.9.4.17 GM_Complex (S25)

The isMaximal operation of GM_Complex is changed to an optional Boolean attribute.

```
GM_Complex::isMaximal[0,1] : Boolean
```

6.10 2D Topology (L5.2D.2d – 2D full topology)

6.10.1 Introduction

This profile consists of a collection of 2D geometric primitives that form a complex and has topological relationships between the primitives. Full Topology refers to data where primitives are disjoint, all boundary primitives exist, and all boundary and coboundary relationships exist between primitives. With boundary and coboundary information adjacency, connectivity, and containment queries are possible. For example with Full Topology a user can begin to ask the questions about evacuation routes from point A to point B if hostile activity is present at location C and location D.

6.10.2 Class diagrams

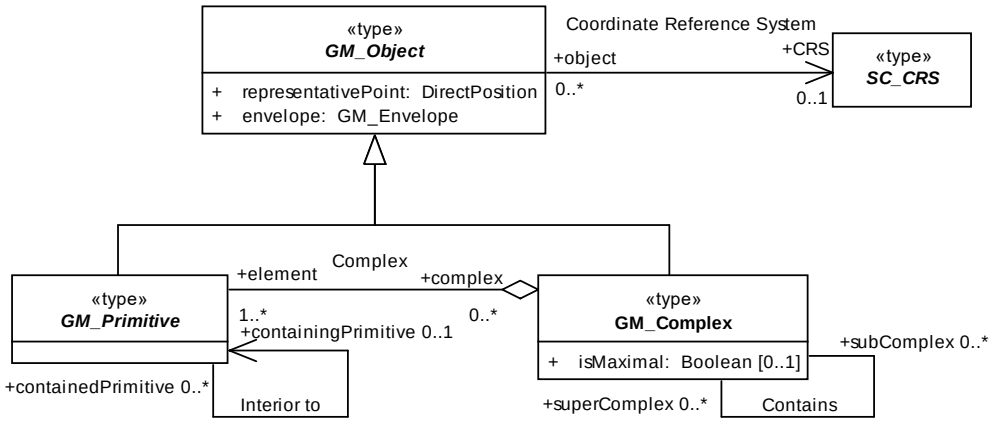


Figure 6.10.1 – Geometry object

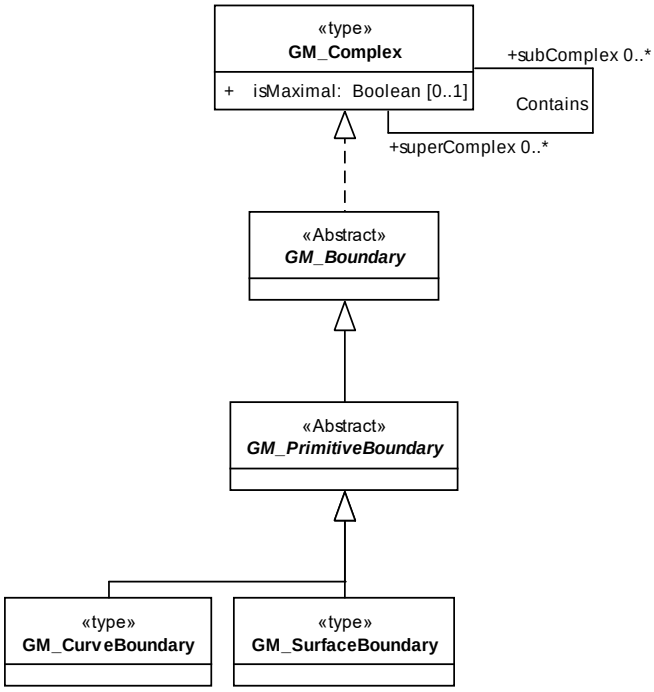


Figure 6.10.2 – Geometry boundary

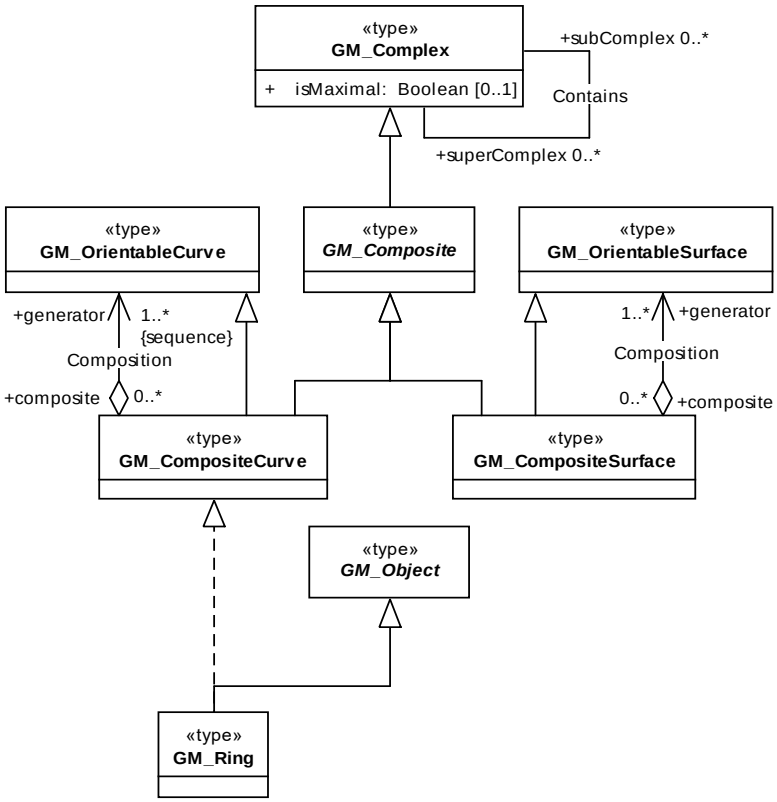


Figure 6.10.3 – Geometry boundary components

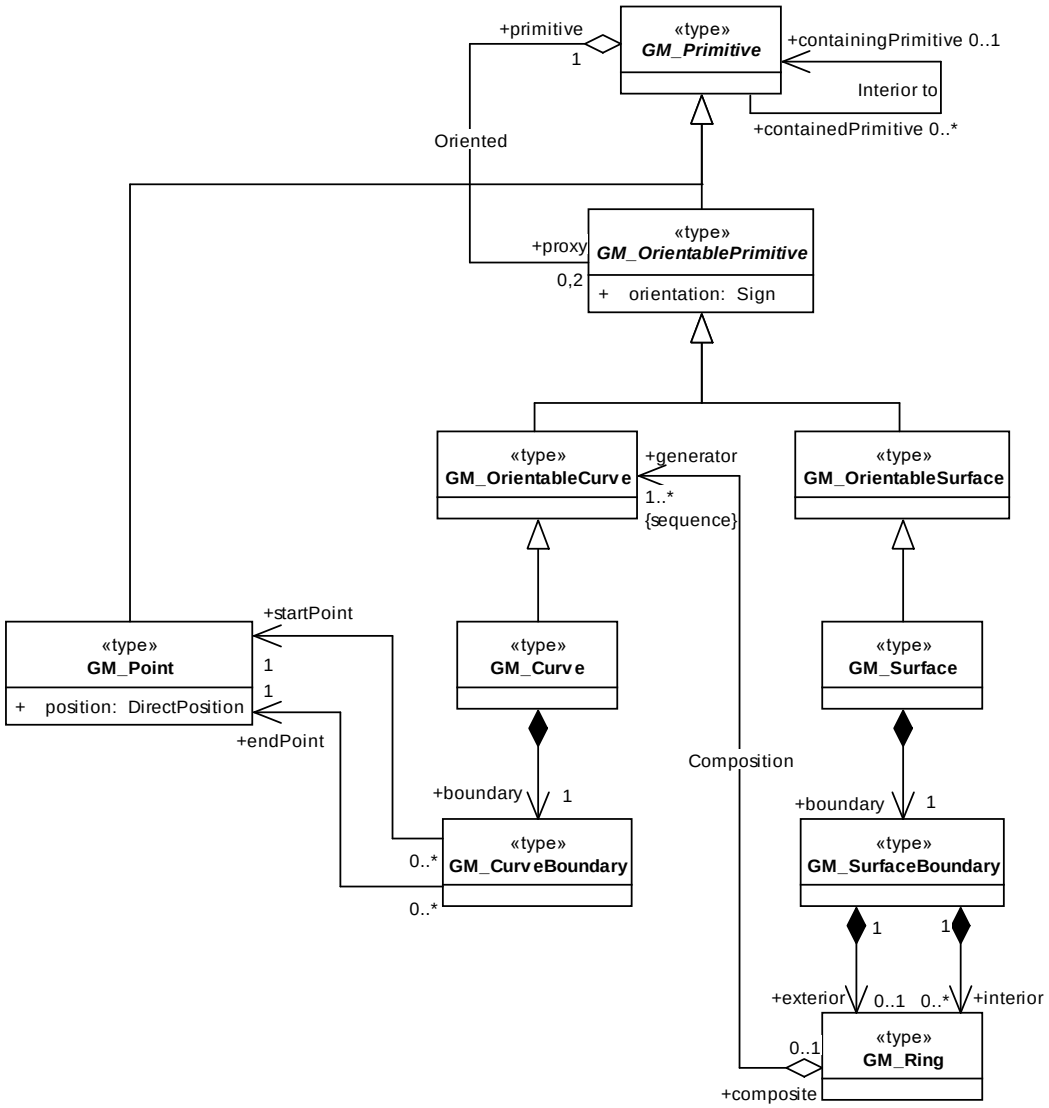


Figure 6.10.4 – Geometry primitive

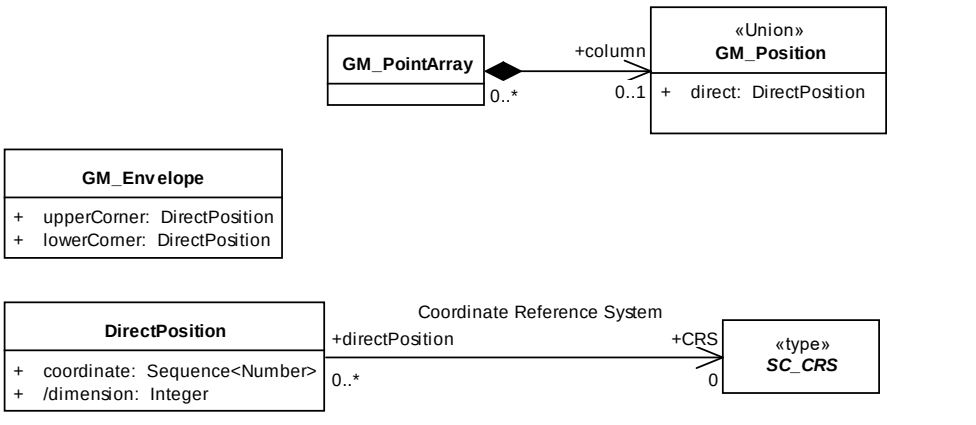


Figure 6.10.5 – Coordinate package

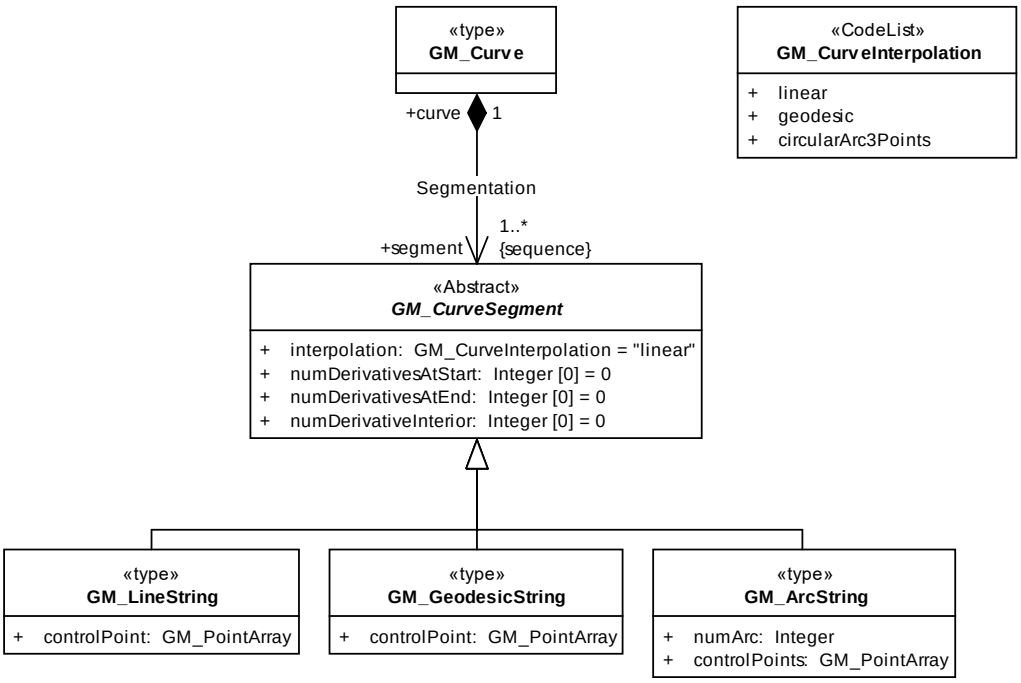


Figure 6.10.6 – Curve segment

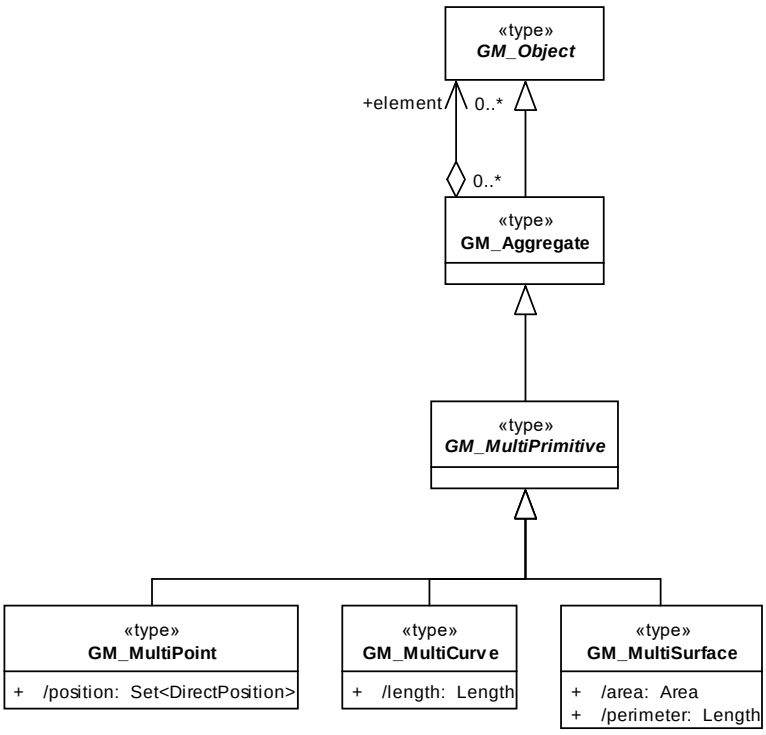


Figure 6.10.7 – Geometry aggregation

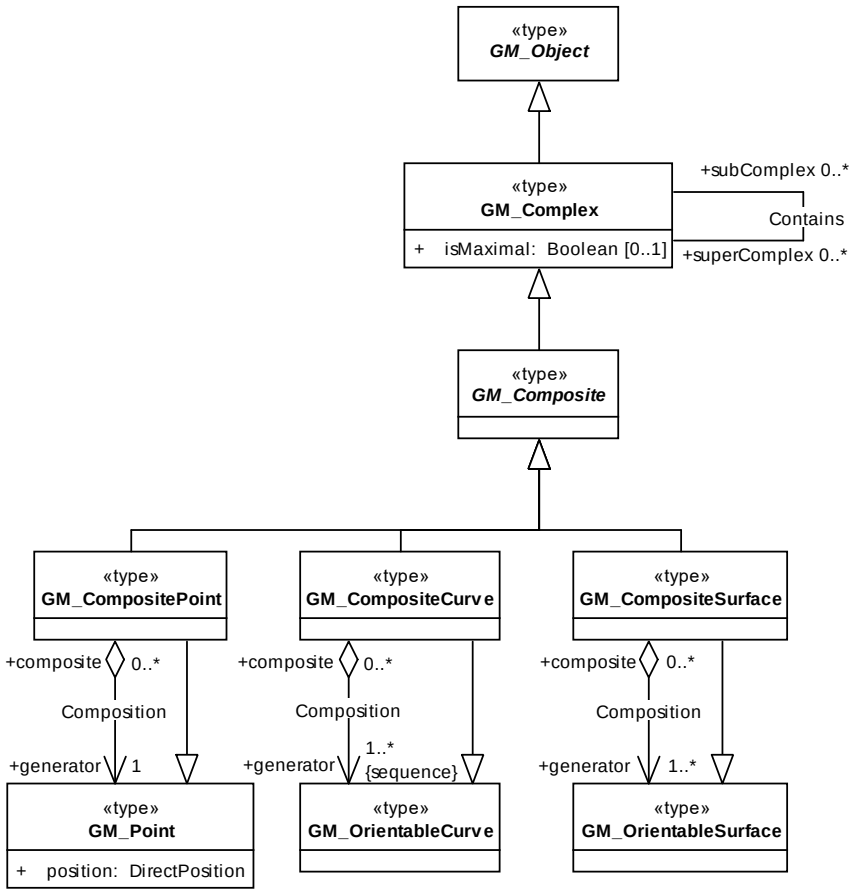


Figure 6.10.8 – Geometry composites

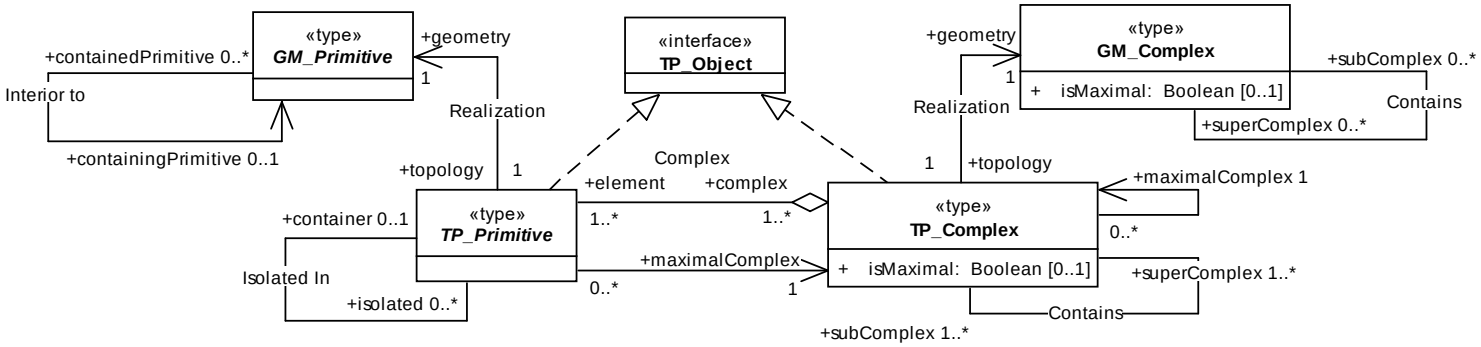


Figure 6.10.9 – Topology object

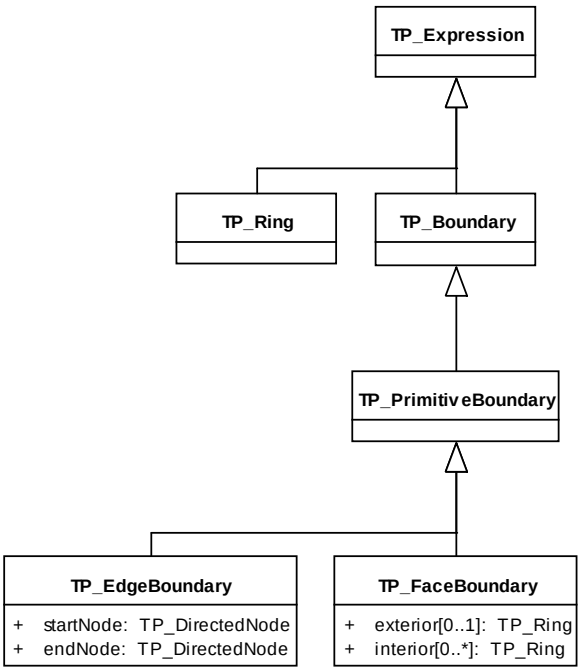


Figure 6.10.10 – Topology boundary

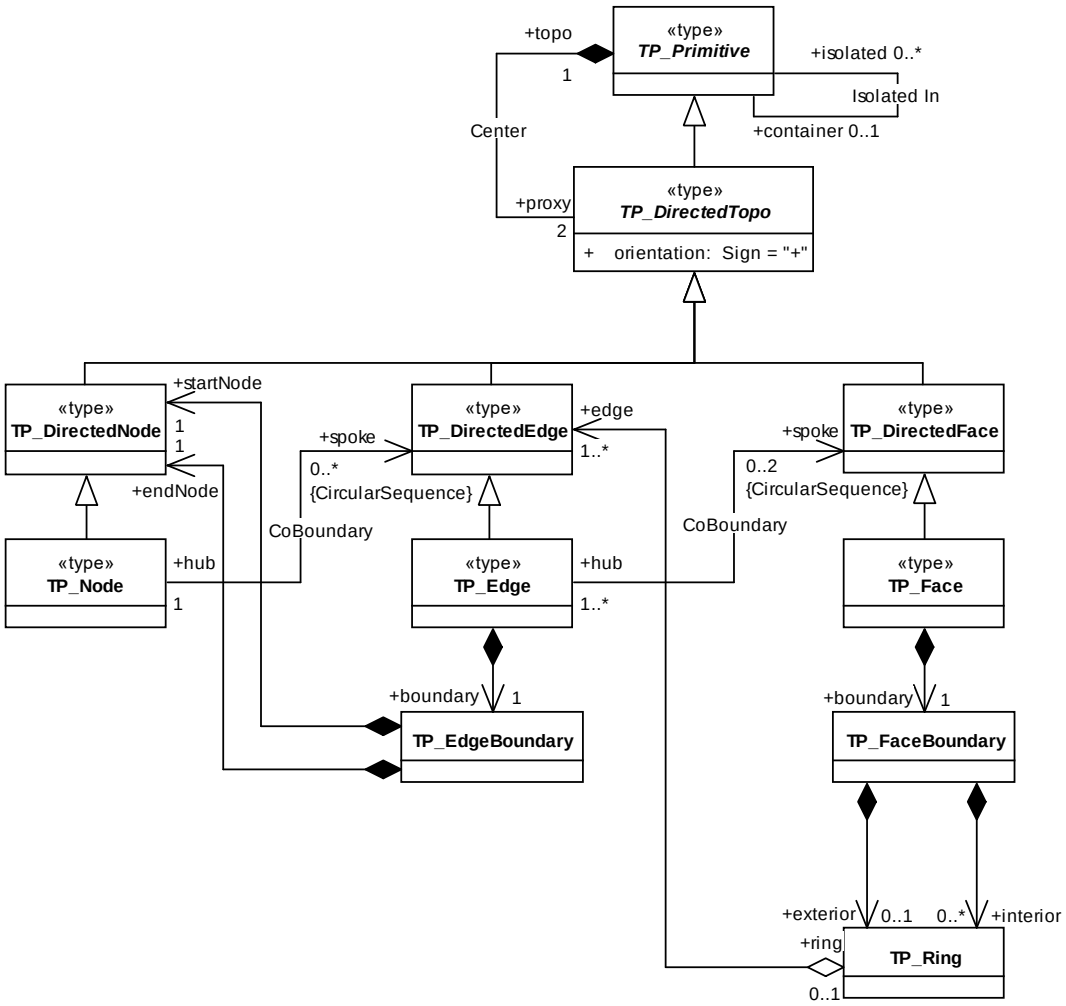


Figure 6.10.11 – Topology primitive

6.10.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.10.4.1(S1), 6.10.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.5 GM_CurveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.10 GM_Primitive – specializations 6.10.4.3(S5), 6.10.4.4(S7), 6.10.4.14(S26)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.10.4.5(S11)
- 6.3.15 GM_OrientableSurface – specialization 6.10.4.6(S12)
- 6.3.16 GM_Curve – specialization 6.10.4.5(S11)
- 6.3.17 GM_Surface – specializations 6.10.4.6(S12), 6.10.4.8(S14)
- 6.4.1 DirectPosition – specialization 6.10.4.9(S17)

- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.10.4.10(S18)
- 6.4.6 GM_PointArray
- 6.4.8 GM_CurveInterpolation – specialization 6.10.4.11(S19)
- 6.4.9 GM_CurveSegment – specializations 6.10.4.12(S20), 0(S13)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.6.2 GM_Complex – specializations 6.10.4.13(S25), 6.10.4.20(S34)
- 6.6.3 GM_Composite
- 6.6.4 GM_CompositePoint
- 6.6.5 GM_CompositeCurve
- 6.6.6 GM_CompositeSurface
- 7.2.2 TP_Object
- 7.3.2 TP_Boundary
- 7.3.4 TP_PrimitiveBoundary
- 7.3.5 TP_EdgeBoundary
- 7.3.6 TP_FaceBoundary
- 7.3.8 TP_Ring
- 7.3.10 TP_Primitive – specializations 6.10.4.14(S26), 6.10.4.15(S27)
- 7.3.11 TP_DirectedTopo
- 7.3.12 TP_Node – specialization 6.10.4.16(S28)
- 7.3.13 TP_DirectedNode
- 7.3.14 TP_Edge – specialization 6.10.4.17(S29)
- 7.3.15 TP_DirectedEdge
- 7.3.16 TP_Face
- 7.3.17 TP_DirectedFace
- 7.3.20 TP_Expression
- 7.4.2 TP_Complex – specializations 6.10.4.18(S32), 6.10.4.19(S33), 6.10.4.20(S34)

6.10.4 Specializations

6.10.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.10.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.10.4.3 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.10.4.4 GM_Primitive (S7)

The InteriorTo association of GM_Primitives is used like the TP_Primitive::IsolatedIn association. The InteriorTo association only associates GM_Primitives with a dimensional difference greater than one. The multiplicity of the superElement association role is constrained from [0..n] to [0..1].

```
GM_Primitive::containingPrimitive[0..1] : GM_Primitive                ([0..n])

context GM_Primitive inv:
containedPrimitive->forAll( p | p.dimension() < self.dimension() - 1)
```

6.10.4.5 GM_OrientableCurve, GM_Curve (S11)

The boundary operation of GM_OrientableCurve is changed in GM_Curve to a composition relationship to GM_CurveBoundary.

```
GM_Curve::boundary[1] : GM_CurveBoundary
```

6.10.4.6 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.10.4.7 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                  ([0,1])
```

6.10.4.8 GM_Surface (S14)

A GM_Surface is described by its boundary; GM_SurfacePatch is not used. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [0].

```
GM_Surface::patch[0] : GM_SurfacePatch                                ([0..n])
```

6.10.4.9 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                 ([0,1])
```

6.10.4.10 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                               ([0,1])
GM_Position::indirect[0] : GM_PointRef                                ([0,1])
```

6.10.4.11 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.10.4.12 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                   ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                      ([0,1])
```

6.10.4.13 GM_Complex (S25)

The isMaximal operation of GM_Complex is changed to an optional Boolean attribute.

```
GM_Complex::isMaximal[0,1] : Boolean
```

6.10.4.14 GM_Primitive, TP_Primitive (S26)

The multiplicity of the geometry role of the Realization association between TP_Primitive and GM_Primitive is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Primitive and TP_Primitive is constrained from [0..n] to [1].

```
TP_Primitive::geometry[1] : GM_Primitive                               ([0,1])
GM_Primitive::topology[1] : TP_Primitive                              ([0..n])
```

6.10.4.15 TP_Primitive (S27)

The association variant of the maximalComplex property of TP_Primitive is used.

6.10.4.16 TP_Node (S28)

The type of the spoke role of the CoBoundary association between TP_Node and TP_DirectedEdge is specialized from a Set to a Circular Sequence.

```
TP_Node::spoke[0..n] : CircularSequence<TP_DirectedEdge>
```

6.10.4.17 TP_Edge (S29)

The multiplicity of the spoke role of the CoBoundary association between TP_Edge and TP_DirectedFace is constrained from [0..n] to [0..2].

```
TP_Edge::spoke[0..2] : CircularSequence<TP_DirectedFace>               ([0..n])
```

6.10.4.18 TP_Complex (S32)

The association variant of the maximalComplex property of TP_Complex is used.

6.10.4.19 TP_Complex (S33)

The isMaximal operation of TP_Complex is changed to an optional Boolean attribute.

```
TP_Complex::isMaximal[0,1] : Boolean
```

6.10.4.20 GM_Complex, TP_Complex (S34)

The multiplicity of the geometry role of the Realization association between TP_Complex and GM_Complex is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Complex and TP_Complex is constrained from [0,1] to [1].

```
TP_Complex::geometry[1] : GM_Complex                                   ([0,1])
GM_Complex::topology[1] : TP_Complex                                   ([0,1])
```

6.11 3D Topology (L5.3D.3d – 3D full topology)

6.11.1 Introduction

This profile consists of a collection of 3D geometric primitives that form a complex and has topological relationships between the primitives. Full Topology refers to data where primitives are disjoint, all boundary primitives exist, and all boundary and coboundary relationships exist between primitives. With boundary and coboundary information adjacency, connectivity, and containment queries are possible. For example with Full Topology a user can begin to ask the questions about evacuation routes from point A to point B if hostile activity is present at location C and location D.

6.11.2 Class diagrams

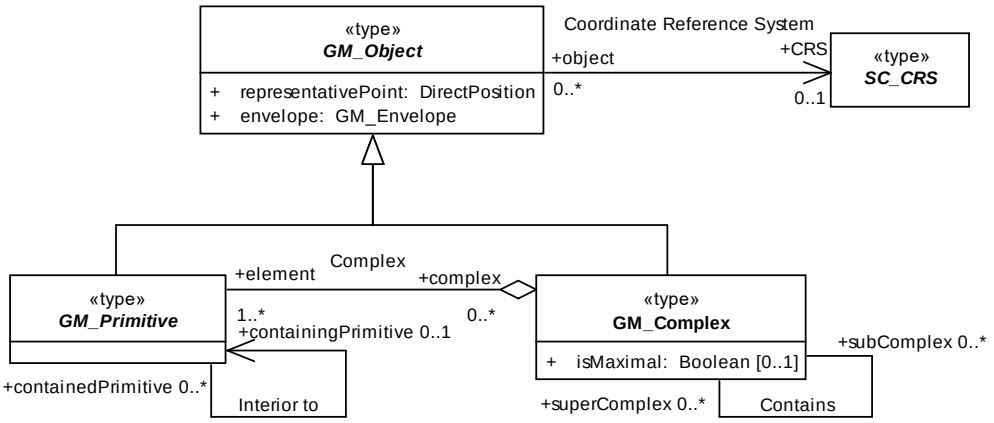


Figure 6.11.1 – Geometry object

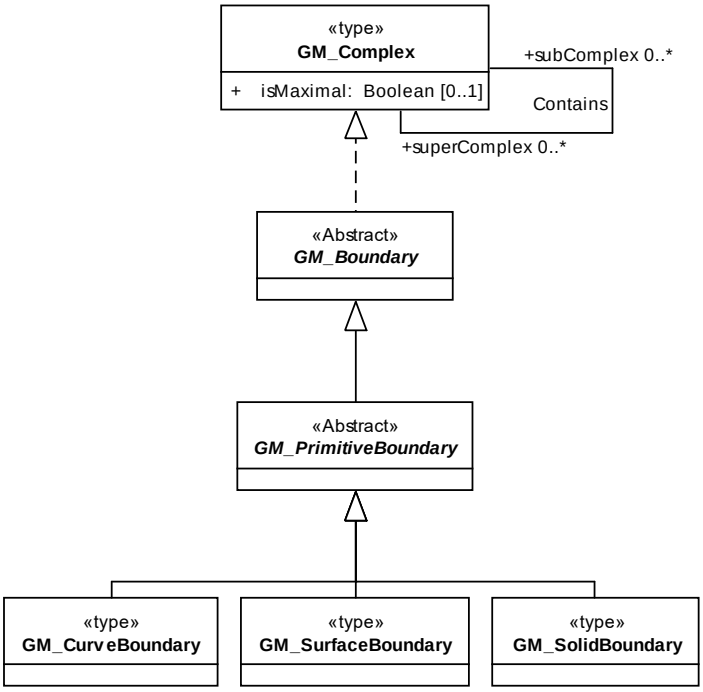


Figure 6.11.2 – Geometry boundary

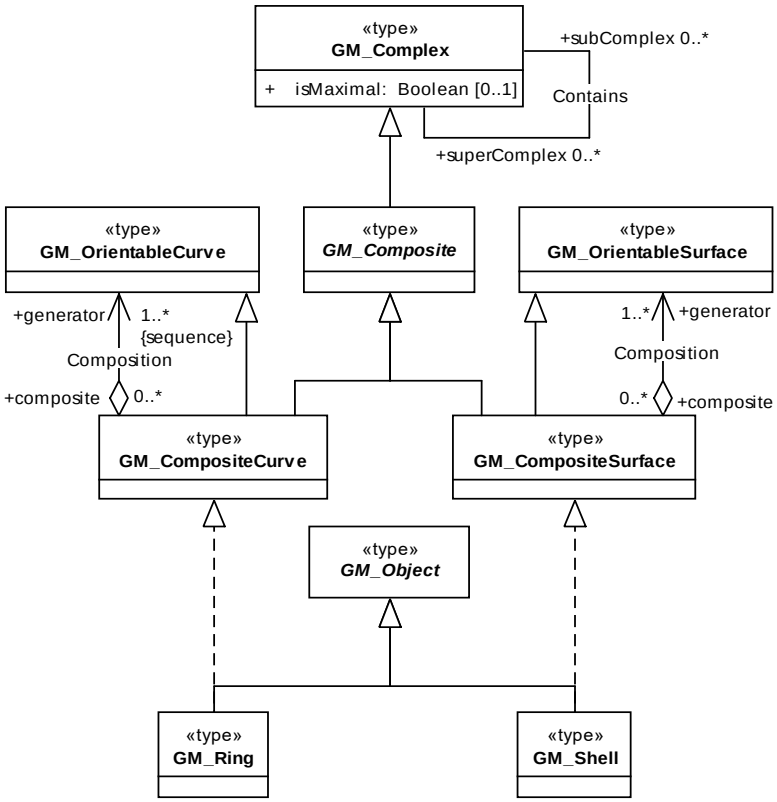


Figure 6.11.3 – Geometry boundary components

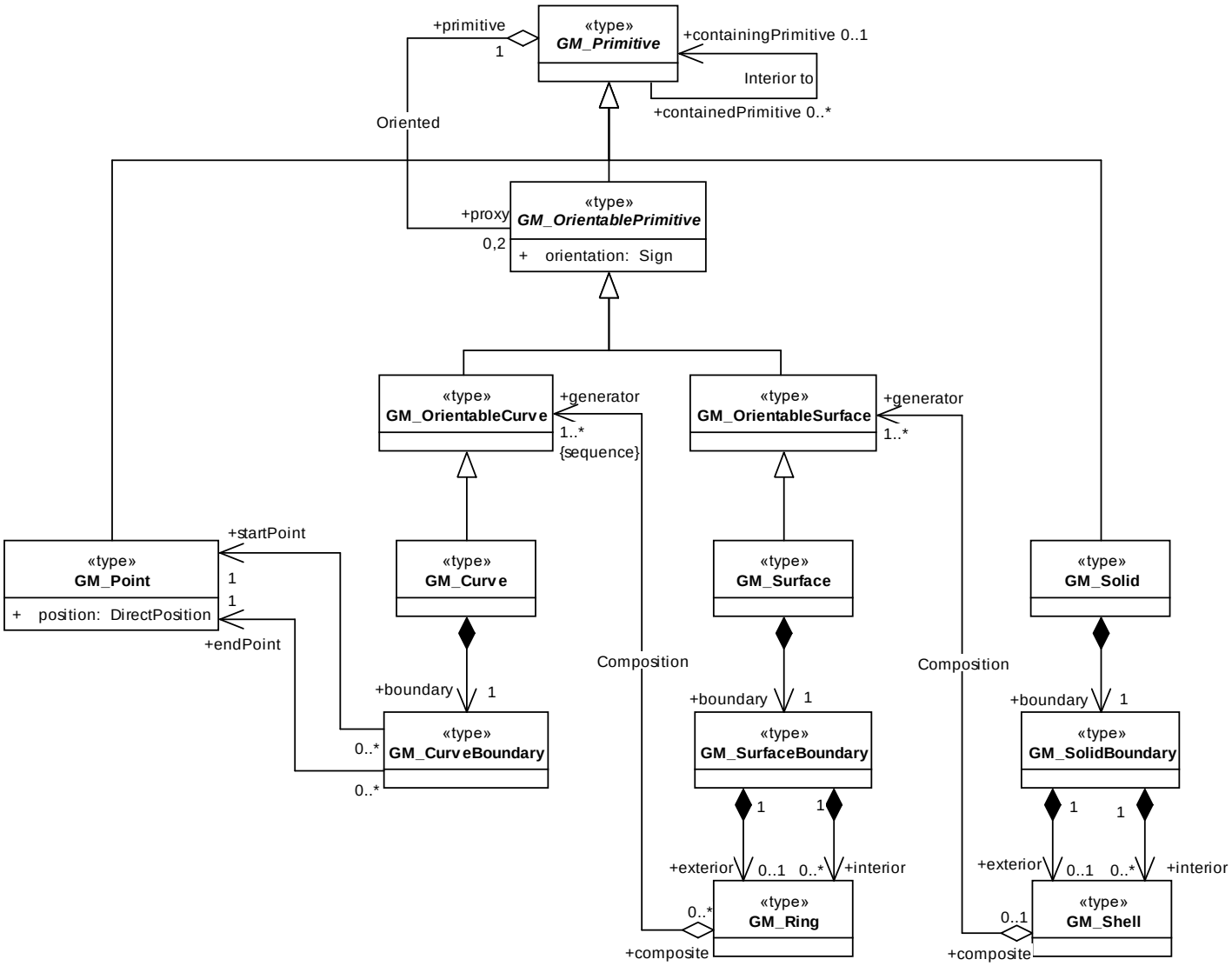


Figure 6.11.4 – Geometry primitive

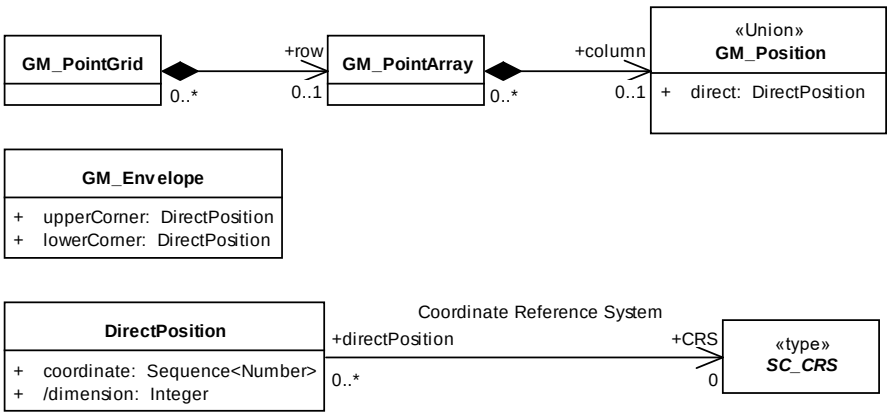


Figure 6.11.5 – Coordinate package

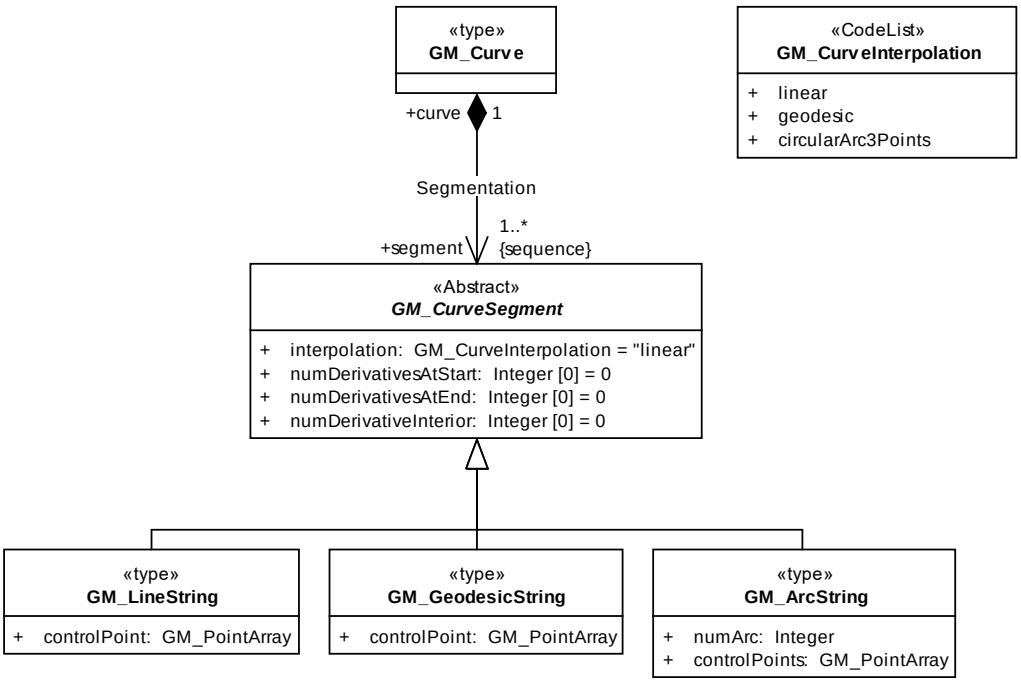


Figure 6.11.6 – Curve segment

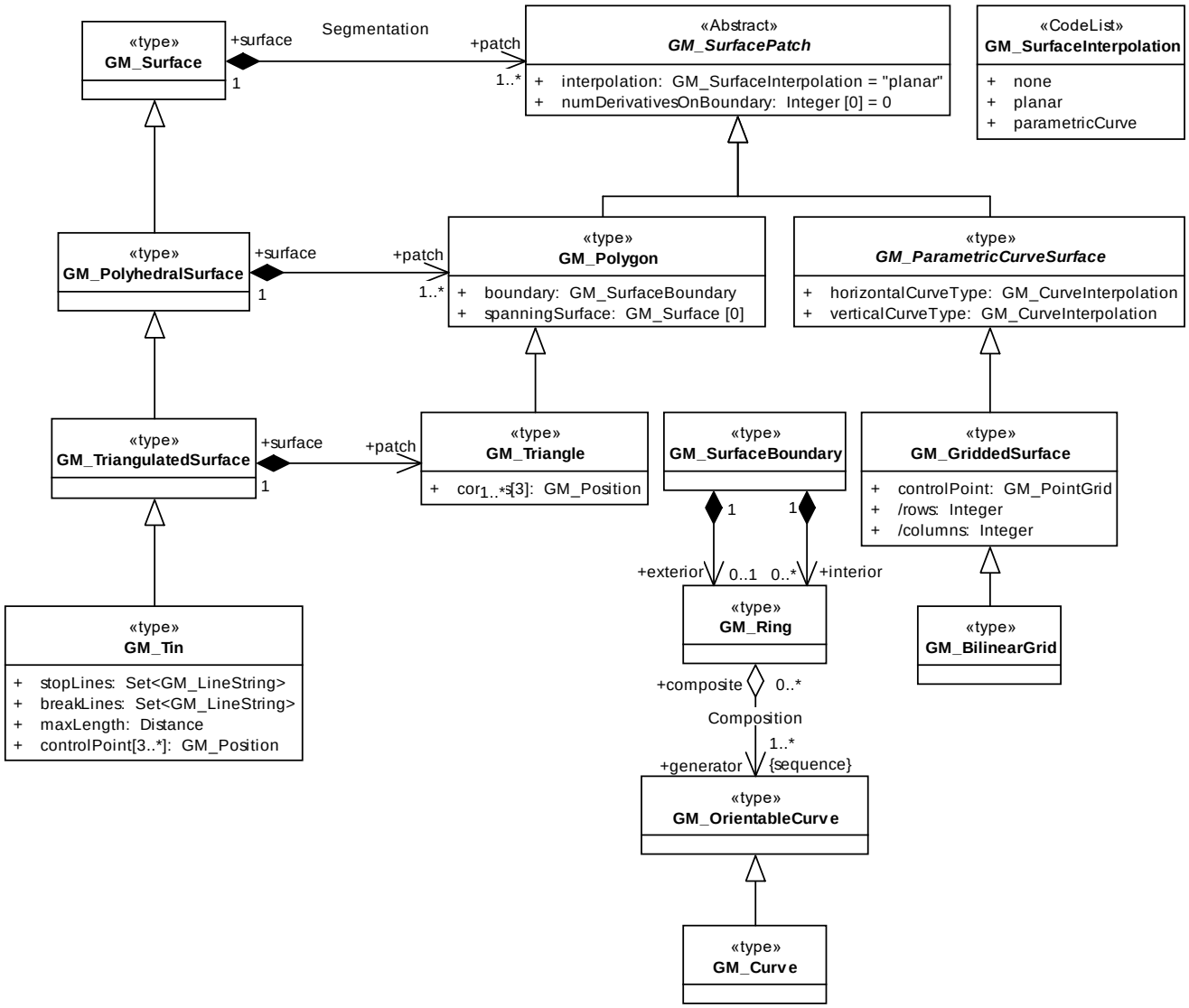


Figure 6.11.7 – Surface patch

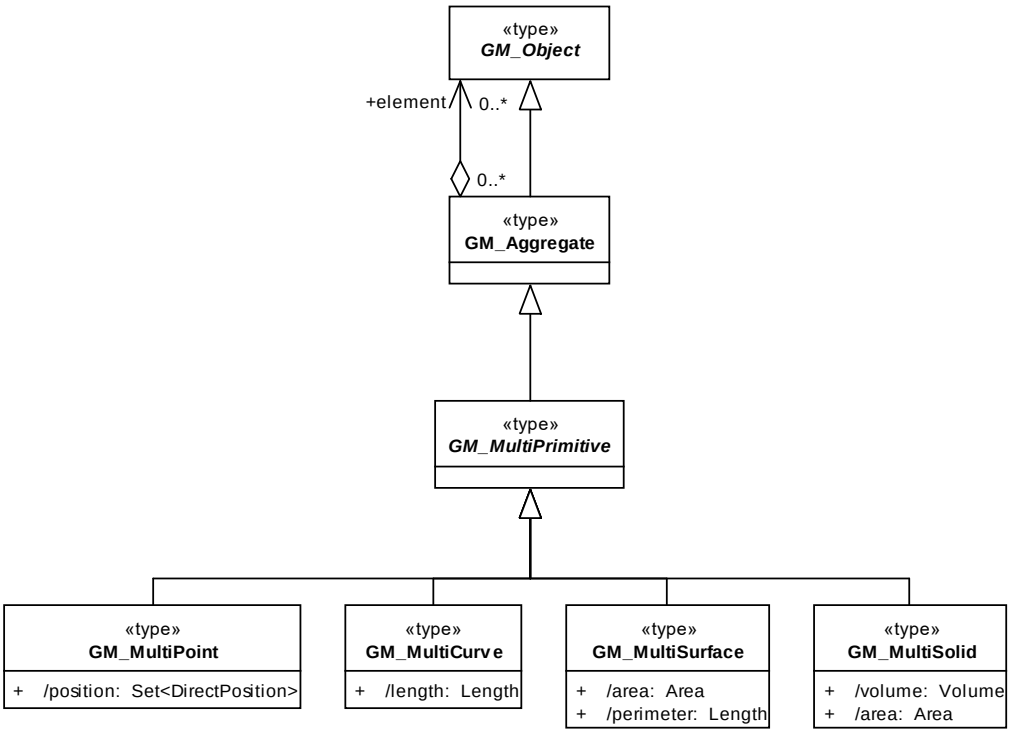


Figure 6.11.8 – Geometry aggregation

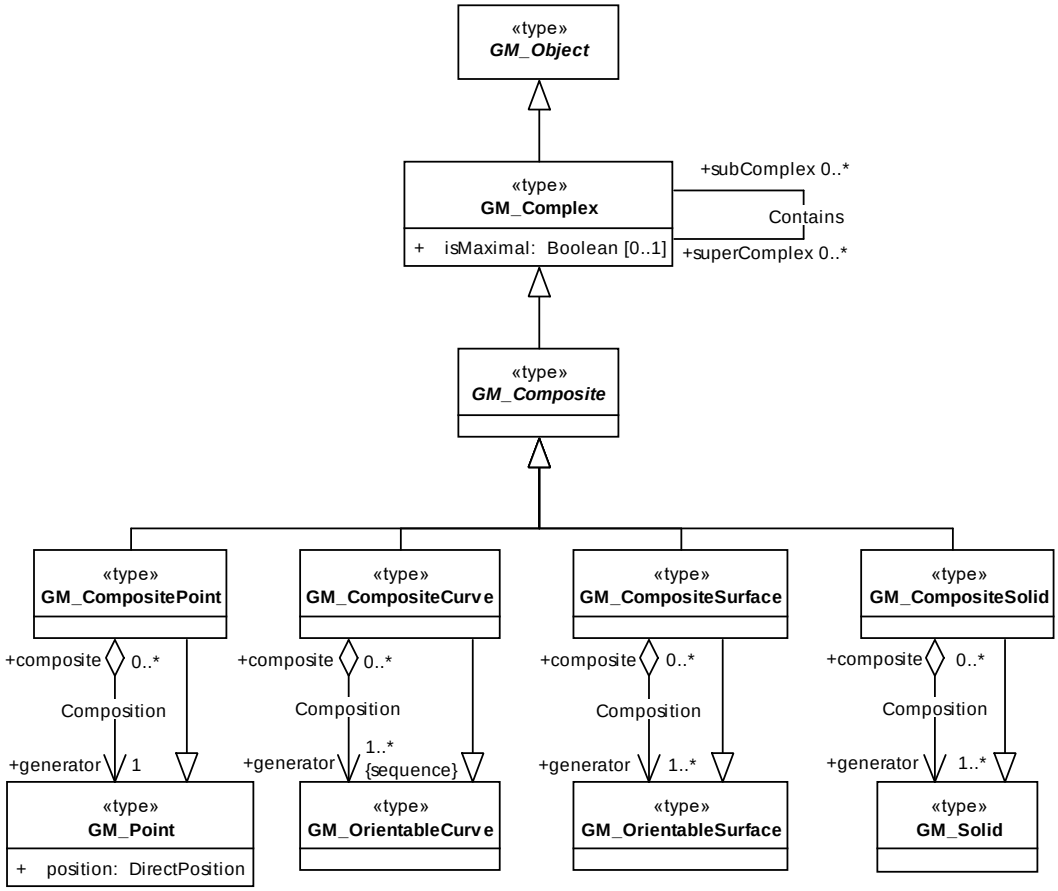


Figure 6.11.9 – Geometry composites

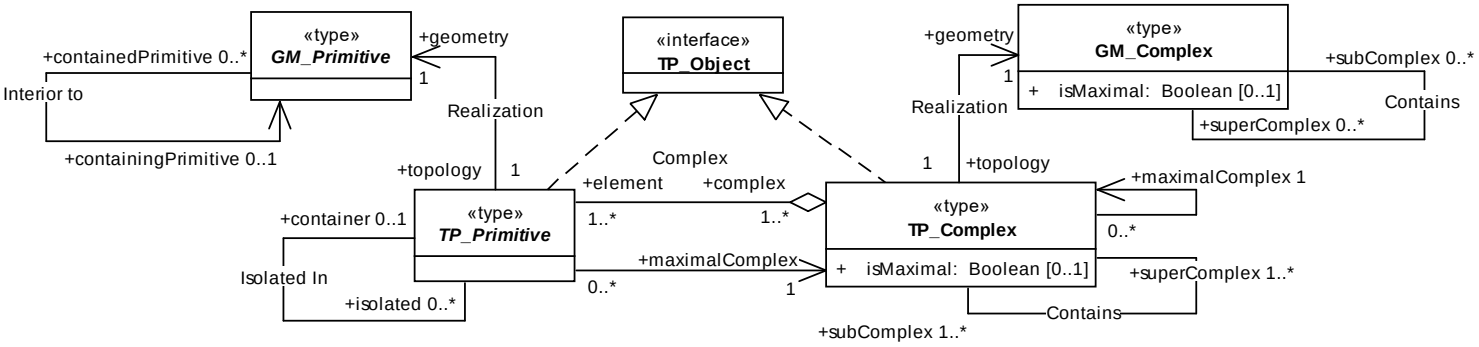


Figure 6.11.10 – Topology object

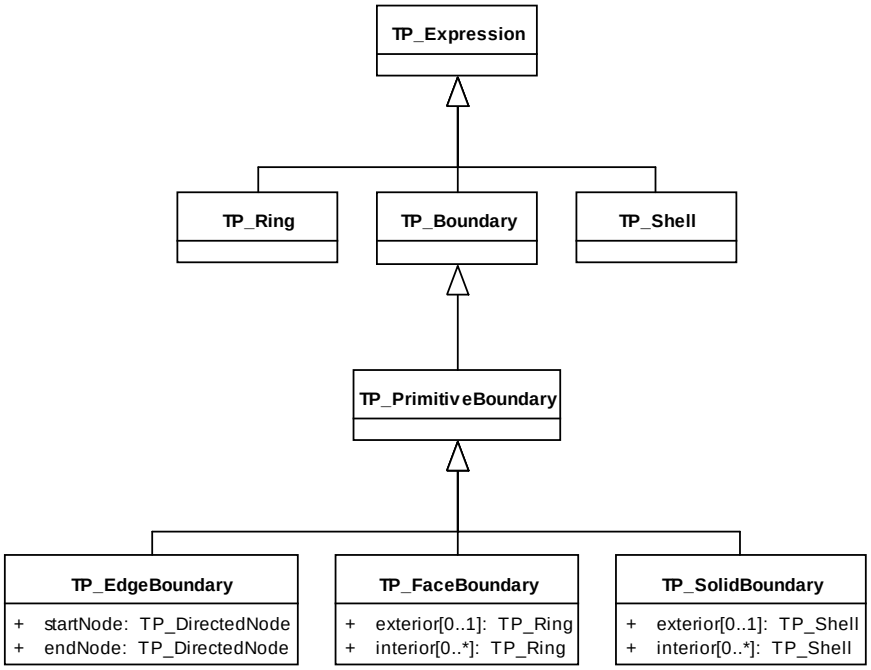


Figure 6.11.11 – Topology boundary

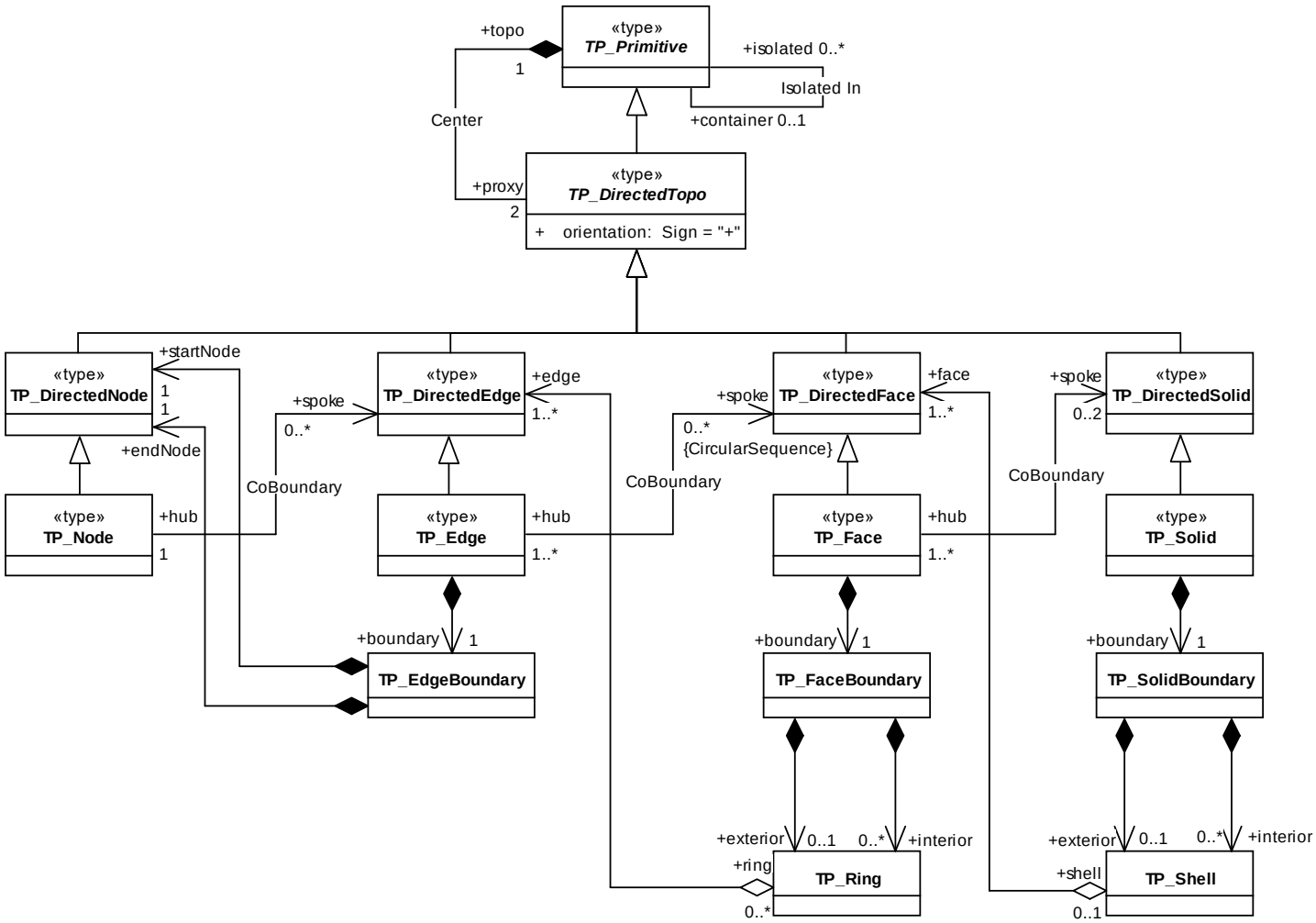


Figure 6.11.12 – Topology primitive

6.11.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object - specializations 6.11.4.1(S1), 6.11.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.5 GM_CurveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.8 GM_Shell
- 6.3.9 GM_SolidBoundary
- 6.3.10 GM_Primitive – specializations 6.11.4.3(S5), 6.11.4.4(S7), 6.11.4.19(S26)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.11.4.5(S11)
- 6.3.15 GM_OrientableSurface – specialization 6.11.4.6(S12)
- 6.3.16 GM_Curve – specialization 6.11.4.5(S11)

- 6.3.17 GM_Surface – specializations 6.11.4.6(S12), 6.11.4.8(S15)
- 6.3.18 GM_Solid – specialization 6.11.4.9(S16)
- 6.4.1 DirectPosition – specialization 6.11.4.10(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.11.4.11(S18)
- 6.4.6 GM_PointArray
- 6.4.6 GM_PointGrid
- 6.4.8 GM_CurveInterpolation – specialization 6.11.4.12(S19)
- 6.4.9 GM_CurveSegment – specializations 0(S13) ,6.11.4.13(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.4.32 GM_SurfaceInterpolation – specialization 6.11.4.14(S21)
- 6.4.34 GM_SurfacePatch – specializations 6.11.4.15(S22), 6.11.4.16(S23)
- 6.4.35 GM_PolyhedralSurface
- 6.4.36 GM_Polygon – specialization 6.11.4.17(S24)
- 6.4.37 GM_TriangulatedSurface
- 6.4.38 GM_Triangle
- 6.4.39 GM_Tin
- 6.4.40 GM_ParametricCurveSurface
- 6.4.41 GM_GriddedSurface
- 6.4.45 GM_BilinearGrid
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.5.7 GM_MultiSolid
- 6.6.2 GM_Complex – specializations 6.11.4.18(S25), 6.11.4.23(S34)
- 6.6.3 GM_Composite
- 6.6.4 GM_CompositePoint
- 6.6.5 GM_CompositeCurve
- 6.6.6 GM_CompositeSurface
- 6.6.7 GM_CompositeSolid
- 7.2.2 TP_Object
- 7.3.2 TP_Boundary
- 7.3.4 TP_PrimitiveBoundary
- 7.3.5 TP_EdgeBoundary
- 7.3.6 TP_FaceBoundary
- 7.3.7 TP_SolidBoundary
- 7.3.8 TP_Ring
- 7.3.9 TP_Shell
- 7.3.10 TP_Primitive – specializations 6.11.4.19(S26), 6.11.4.20(S27)
- 7.3.11 TP_DirectedTopo

- 7.3.12 TP_Node
- 7.3.13 TP_DirectedNode
- 7.3.14 TP_Edge
- 7.3.15 TP_DirectedEdge
- 7.3.16 TP_Face
- 7.3.17 TP_DirectedFace
- 7.3.18 TP_Solid
- 7.3.19 TP_DirectedSolid
- 7.3.20 TP_Expression
- 7.4.2 TP_Complex – specializations 6.11.4.21(S32), 6.11.4.22(S33), 6.11.4.23(S34)

6.11.4 Specializations

6.11.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.11.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.11.4.3 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.11.4.4 GM_Primitive (S7)

The InteriorTo association of GM_Primitives is used like the TP_Primitive::IsolatedIn association. The InteriorTo association only associates GM_Primitives with a dimensional difference greater than one. The multiplicity of the superElement association role is constrained from [0..n] to [0..1].

```
GM_Primitive::containingPrimitive[0..1] : GM_Primitive                    ([0..n])

context GM_Primitive inv:
containedPrimitive->forAll( p | p.dimension() < self.dimension() - 1)
```

6.11.4.5 GM_OrientableCurve, GM_Curve (S11)

The boundary operation of GM_OrientableCurve is changed in GM_Curve to a composition relationship to GM_CurveBoundary.

```
GM_Curve::boundary[1] : GM_CurveBoundary
```

6.11.4.6 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.11.4.7 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                        ([0,1])
```

6.11.4.8 GM_Surface (S15)

The interior of a GM_Surface patch is described by GM_SurfacePatches. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [1..n].

```
GM_Surface::patch[1..n] : GM_SurfacePatch                    ([0..n])
```

6.11.4.9 GM_Solid (S16)

The boundary operation of GM_Solid is changed to a composition relationship to GM_SolidBoundary.

```
GM_Solid::boundary[1] : GM_SolidBoundary
```

6.11.4.10 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS        ([0,1])
```

6.11.4.11 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                      ([0,1])
GM_Position::indirect[0] : GM_PointRef                       ([0,1])
```

6.11.4.12 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.11.4.13 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer          ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer         ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer            ([0,1])
```

6.11.4.14 GM_SurfaceInterpolation (S21)

The GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.

```
GM_SurfaceInterpolation::
none
planar
parametricCurve
```

6.11.4.15 GM_SurfacePatch (S22)

The multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained from [0,1] to [1].

```
GM_SurfacePatch::surface[1] : GM_Surface                     ([0,1])
```

6.11.4.16 GM_SurfacePatch (S23)

The multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained from [0,1] to [0].

```
GM_SurfacePatch::numDerivativesOnBoundary[0] : Integer       ([0,1])
```

6.11.4.17 GM_Polygon (S24)

The multiplicity of the spanningSurface attribute of GM_Polygon is constrained from [0..1] to [0].

```
GM_Polygon::spanningSurface[0] : GM_Surface                    ([0,1])
```

6.11.4.18 GM_Complex (S25)

The isMaximal operation of GM_Complex is changed to an optional Boolean attribute.

```
GM_Complex::isMaximal[0,1] : Boolean
```

6.11.4.19 GM_Primitive, TP_Primitive (S26)

The multiplicity of the geometry role of the Realization association between TP_Primitive and GM_Primitive is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Primitive and TP_Primitive is constrained from [0..n] to [1].

```
TP_Primitive::geometry[1] : GM_Primitive                       ([0,1])
GM_Primitive::topology[1] : TP_Primitive                      ([0..n])
```

6.11.4.20 TP_Primitive (S27)

The association variant of the maximalComplex property of TP_Primitive is used.

6.11.4.21 TP_Complex (S32)

The association variant of the maximalComplex property of TP_Complex is used.

6.11.4.22 TP_Complex (S33)

The isMaximal operation of TP_Complex is changed to an optional Boolean attribute.

```
TP_Complex::isMaximal[0,1] : Boolean
```

6.11.4.23 GM_Complex, TP_Complex (S34)

The multiplicity of the geometry role of the Realization association between TP_Complex and GM_Complex is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Complex and TP_Complex is constrained from [0,1] to [1].

```
TP_Complex::geometry[1] : GM_Complex                           ([0,1])
GM_Complex::topology[1] : TP_Complex                           ([0,1])
```

6.12 2D Tessellation (L6.2D.2d – 2D partitioned space)

6.12.1 Introduction

This profile consists of a collection of 2D geometric primitives that form a tessellation of a surface, i.e. a partitioning of the surface. Partitioned Space refers to data which is equivalent to Full Topology with the added requirement that all “blank” space is filled. This becomes important when assessing cross country movement and soil information. This corresponds to VRF Level 3.

6.12.2 Class diagrams

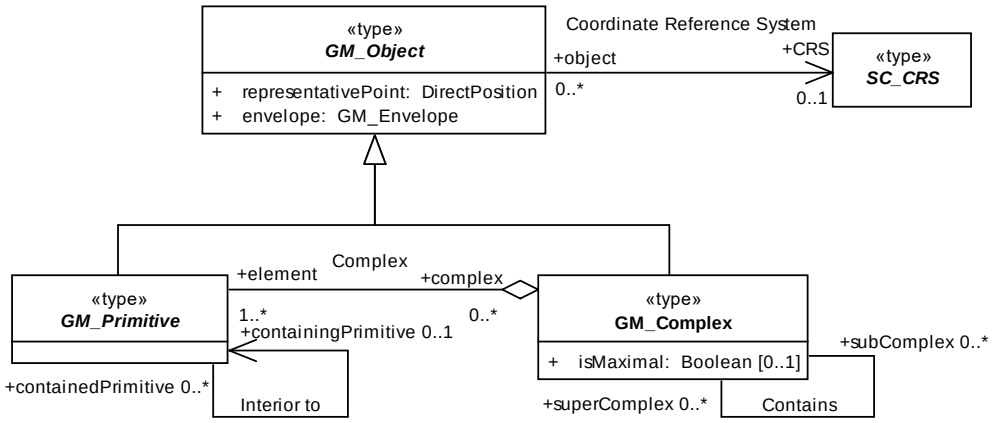


Figure 6.12.1 – Geometry object

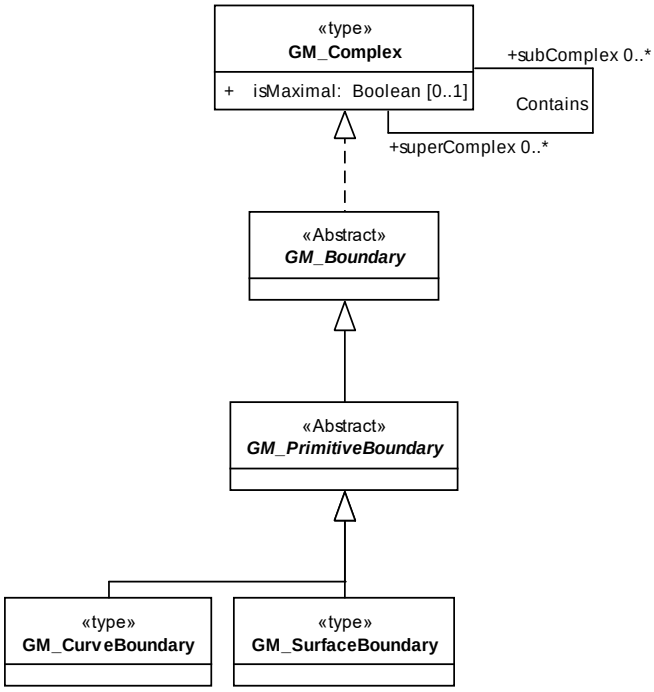


Figure 6.12.2 – Geometry boundary

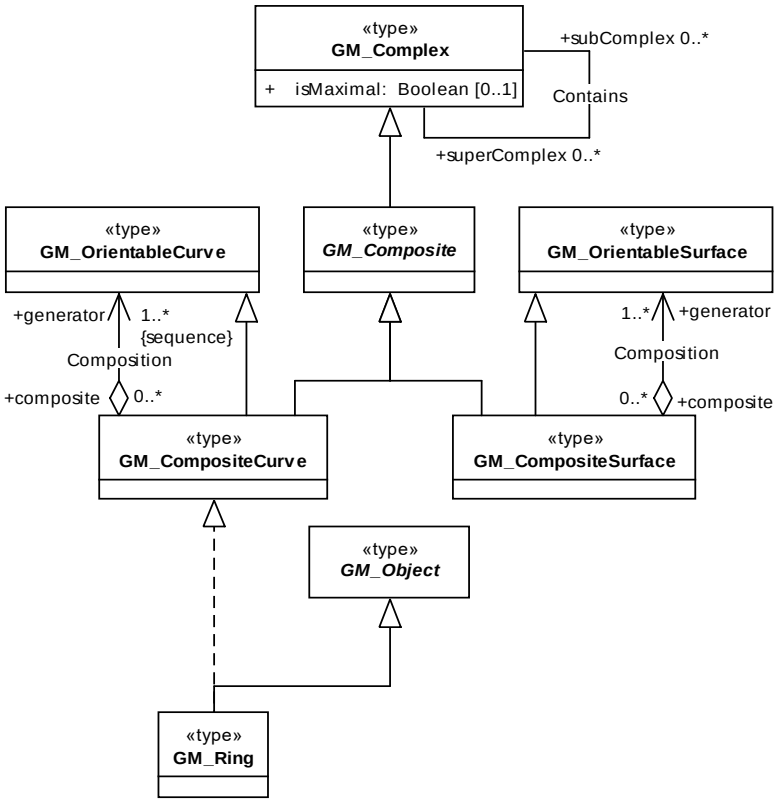


Figure 6.12.3 – Geometry boundary components

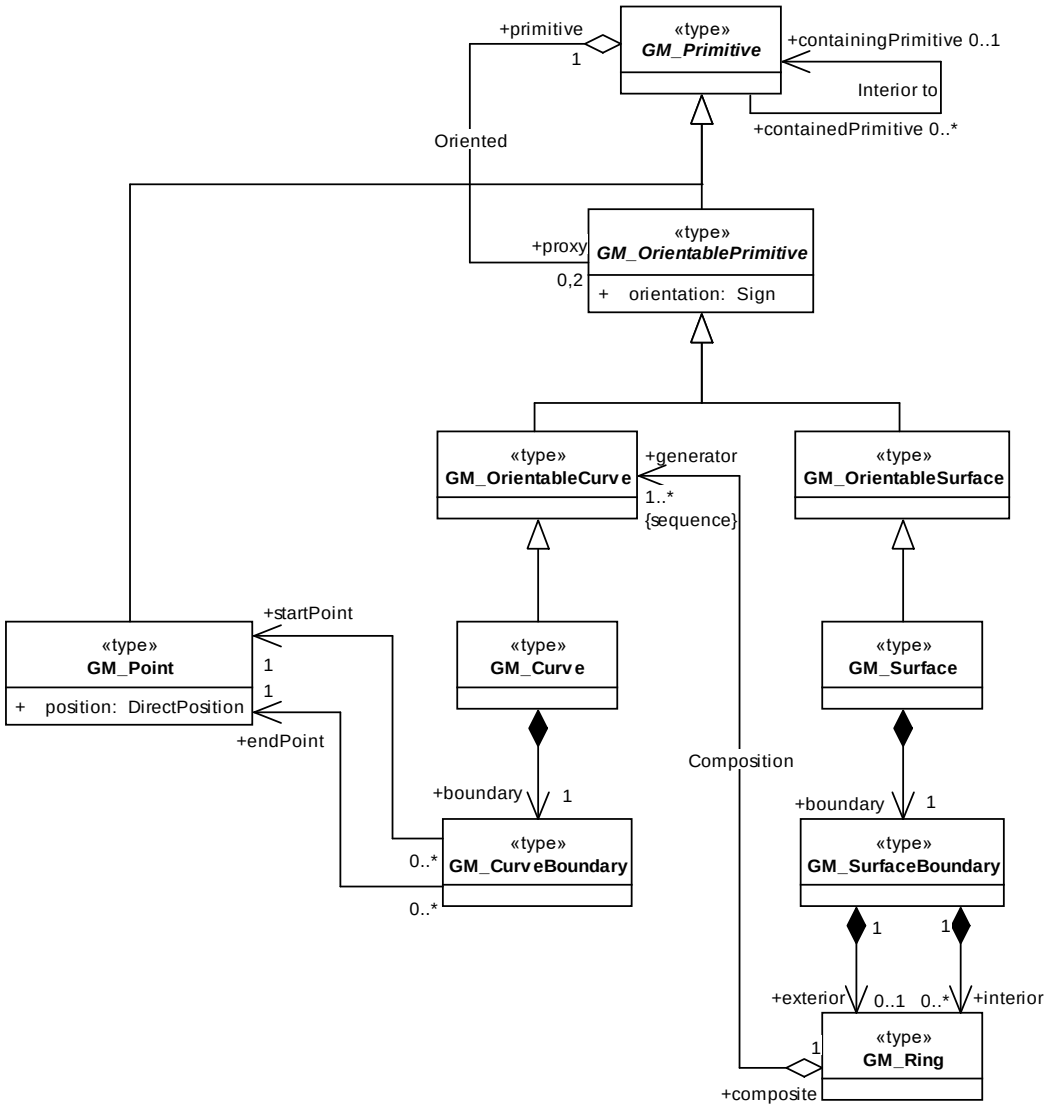


Figure 6.12.4 – Geometry primitive

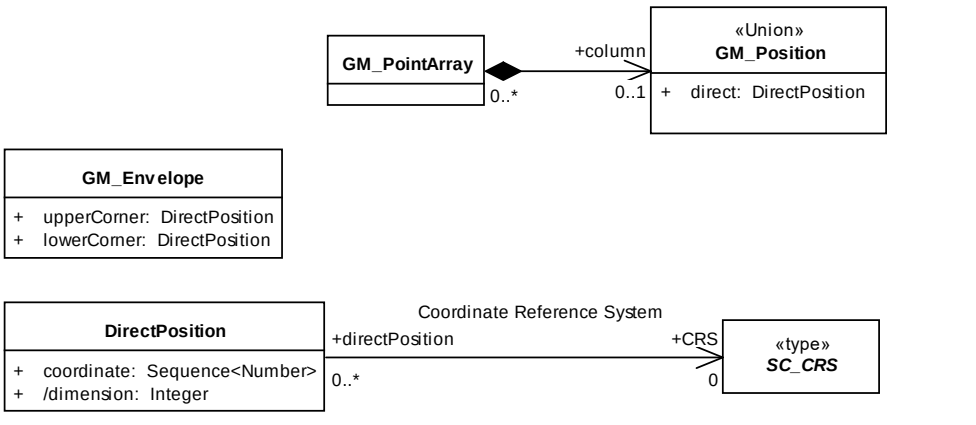


Figure 6.12.5 – Coordinate package

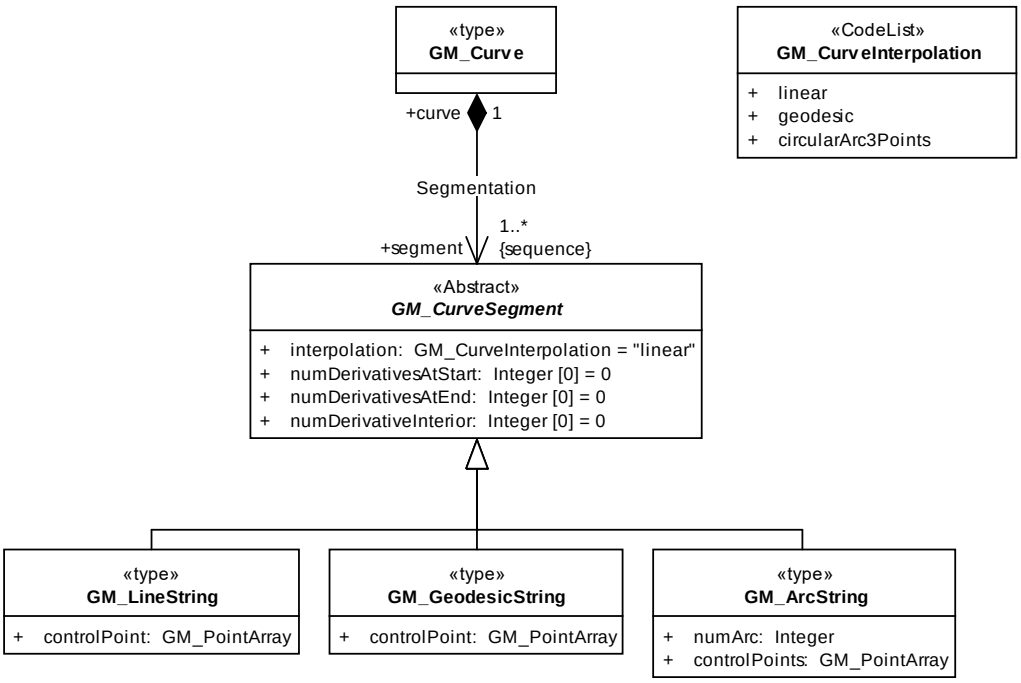


Figure 6.12.6 – Curve segment

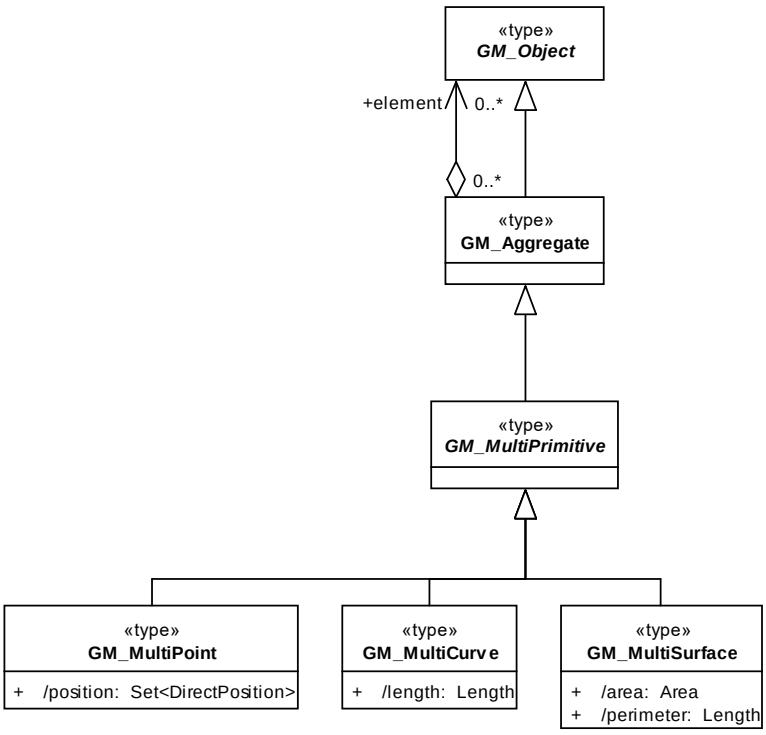


Figure 6.12.7 – Geometry aggregation

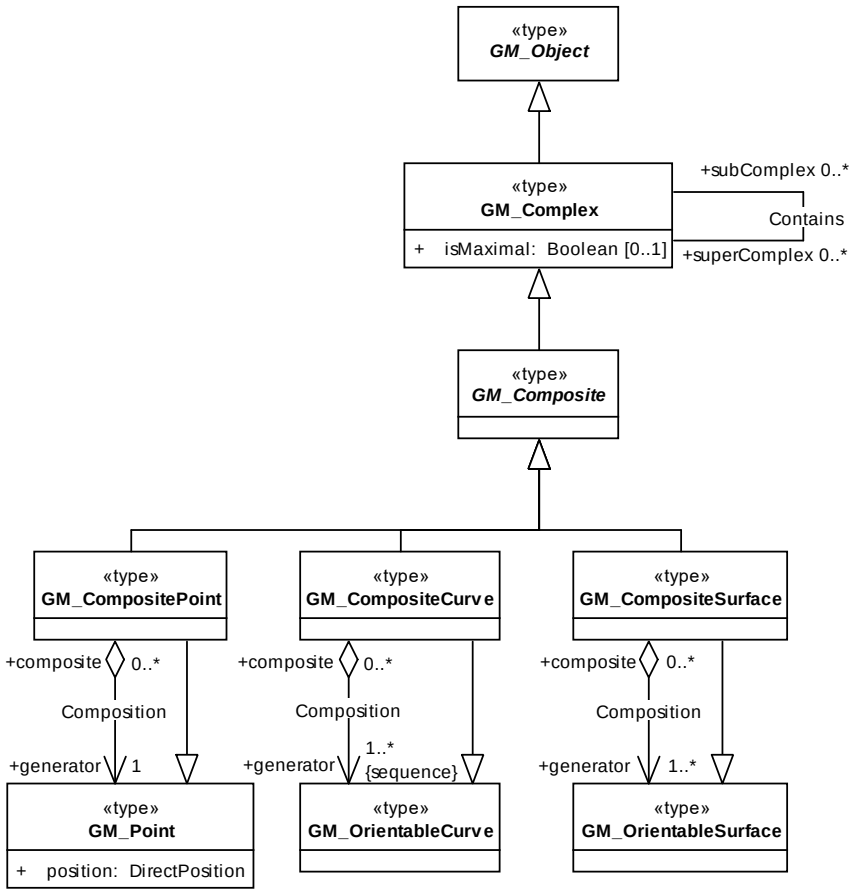


Figure 6.12.8 – Geometry composites

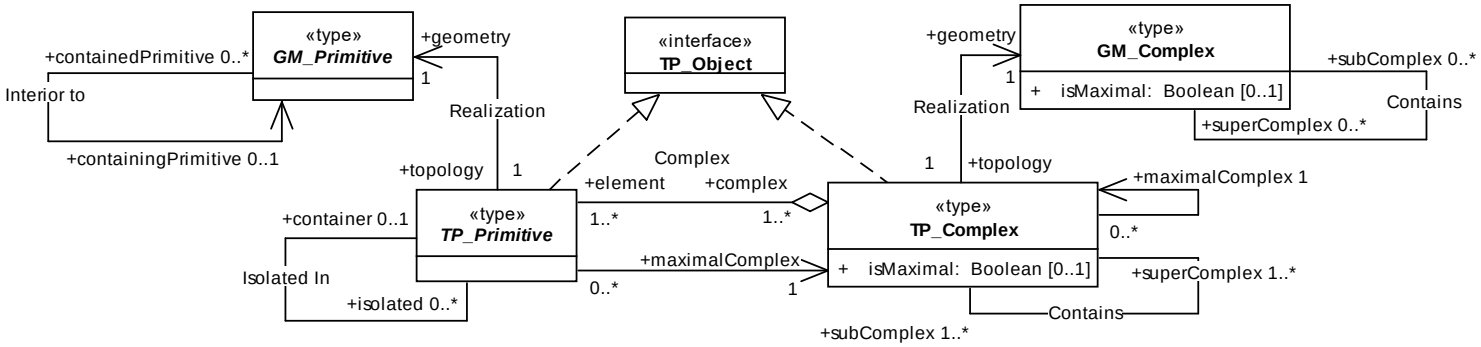


Figure 6.12.9 – Topology object

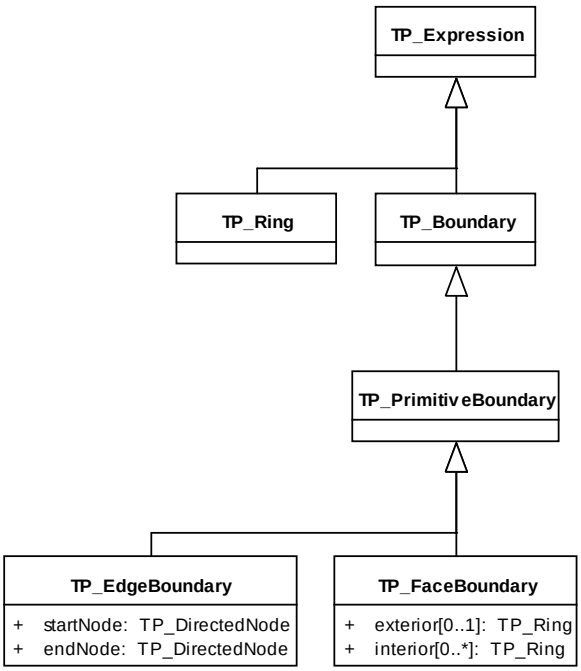


Figure 6.12.10 – Topology boundary

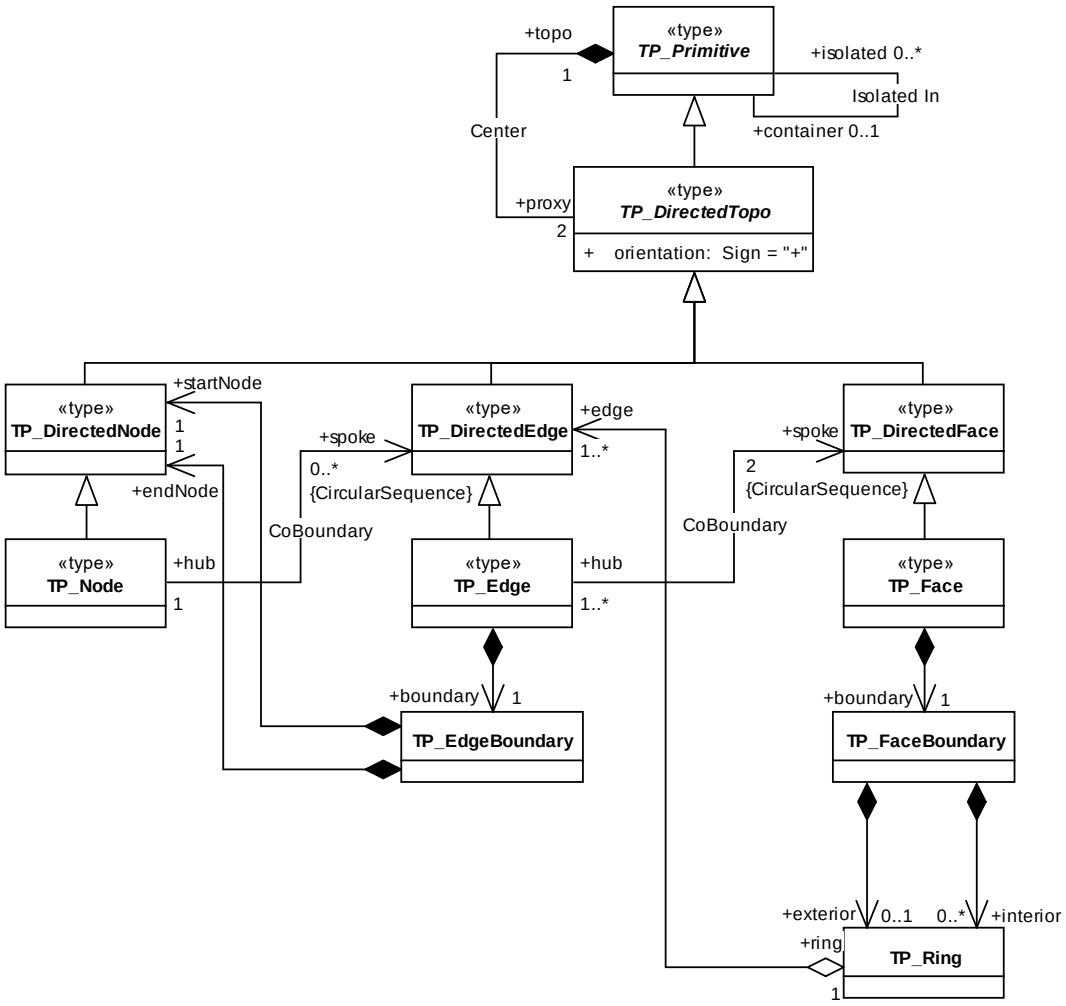


Figure 6.12.11 – Topology primitive

6.12.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.12.4.1(S1), 6.12.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.5 GM_CurveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.10 GM_Primitive – specializations 6.12.4.3(S5), 6.12.4.4(S7), 6.12.4.14(S26)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.12.4.5(S11)
- 6.3.15 GM_OrientableSurface – specialization 6.12.4.6(S12)
- 6.3.16 GM_Curve – specialization 6.12.4.5(S11)
- 6.3.17 GM_Surface – specializations 6.12.4.6(S12), 6.12.4.8(S14)
- 6.4.1 DirectPosition – specialization 6.12.4.9(S17)

- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.12.4.10(S18)
- 6.4.6 GM_PointArray
- 6.4.8 GM_CurveInterpolation – specialization 6.12.4.11(S19)
- 6.4.9 GM_CurveSegment – specializations 0(S13) ,6.12.4.12(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.6.2 GM_Complex – specializations 6.12.4.13(S25), 6.12.4.20(S34)
- 6.6.3 GM_Composite
- 6.6.4 GM_CompositePoint
- 6.6.5 GM_CompositeCurve
- 6.6.6 GM_CompositeSurface
- 7.2.2 TP_Object
- 7.3.2 TP_Boundary
- 7.3.4 TP_PrimitiveBoundary
- 7.3.5 TP_EdgeBoundary
- 7.3.6 TP_FaceBoundary
- 7.3.8 TP_Ring
- 7.3.10 TP_Primitive – specializations 6.12.4.14(S26), 6.12.4.15(S27)
- 7.3.11 TP_DirectedTopo
- 7.3.12 TP_Node – specialization 6.12.4.16(S28)
- 7.3.13 TP_DirectedNode
- 7.3.14 TP_Edge – specialization 6.12.4.17(S30)
- 7.3.15 TP_DirectedEdge
- 7.3.16 TP_Face
- 7.3.17 TP_DirectedFace
- 7.3.20 TP_Expression
- 7.4.2 TP_Complex – specializations 6.12.4.18(S32), 6.12.4.19(S33), 6.12.4.20(S34)

6.12.4 Specializations

6.12.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.12.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.12.4.3 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.12.4.4 GM_Primitive (S7)

The InteriorTo association of GM_Primitives is used like the TP_Primitive::IsolatedIn association. The InteriorTo association only associates GM_Primitives with a dimensional difference greater than one. The multiplicity of the superElement association role is constrained from [0..n] to [0..1].

```
GM_Primitive::containingPrimitive[0..1] : GM_Primitive                ([0..n])

context GM_Primitive inv:
containedPrimitive->forAll( p | p.dimension() < self.dimension() - 1)
```

6.12.4.5 GM_OrientableCurve, GM_Curve (S11)

The boundary operation of GM_OrientableCurve is changed in GM_Curve to a composition relationship to GM_CurveBoundary.

```
GM_Curve::boundary[1] : GM_CurveBoundary
```

6.12.4.6 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.12.4.7 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                  ([0,1])
```

6.12.4.8 GM_Surface (S14)

A GM_Surface is described by its boundary; GM_SurfacePatch is not used. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [0].

```
GM_Surface::patch[0] : GM_SurfacePatch                                ([0..n])
```

6.12.4.9 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS                 ([0,1])
```

6.12.4.10 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                               ([0,1])
GM_Position::indirect[0] : GM_PointRef                                ([0,1])
```

6.12.4.11 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.12.4.12 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer                    ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer                   ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer                      ([0,1])
```

6.12.4.13 GM_Complex (S25)

The isMaximal operation of GM_Complex is changed to an optional Boolean attribute.

```
GM_Complex::isMaximal[0,1] : Boolean
```

6.12.4.14 GM_Primitive, TP_Primitive (S26)

The multiplicity of the geometry role of the Realization association between TP_Primitive and GM_Primitive is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Primitive and TP_Primitive is constrained from [0..n] to [1].

```
TP_Primitive::geometry[1] : GM_Primitive                               ([0,1])
GM_Primitive::topology[1] : TP_Primitive                              ([0..n])
```

6.12.4.15 TP_Primitive (S27)

The association variant of the maximalComplex property of TP_Primitive is used.

6.12.4.16 TP_Node (S28)

The type of the spoke role of the CoBoundary association between TP_Node and TP_DirectedEdge is specialized from a Set to a Circular Sequence.

```
TP_Node::spoke[0..n] : CircularSequence<TP_DirectedEdge>
```

6.12.4.17 TP_Edge (S30)

The multiplicity of the spoke role of the CoBoundary association between TP_Edge and TP_DirectedFace is constrained from [0..n] to [2].

```
TP_Edge::spoke[2] : CircularSequence<TP_DirectedFace>                  ([0..n])
```

6.12.4.18 TP_Complex (S32)

The association variant of the maximalComplex property of TP_Complex is used.

6.12.4.19 TP_Complex (S33)

The isMaximal operation of TP_Complex is changed to an optional Boolean attribute.

```
TP_Complex::isMaximal[0,1] : Boolean
```

6.12.4.20 GM_Complex, TP_Complex (S34)

The multiplicity of the geometry role of the Realization association between TP_Complex and GM_Complex is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Complex and TP_Complex is constrained from [0,1] to [1].

```
TP_Complex::geometry[1] : GM_Complex                                   ([0,1])
GM_Complex::topology[1] : TP_Complex                                   ([0,1])
```

6.13 3D Tessellation (L6.3D.3d – 3D partitioned space)

6.13.1 Introduction

This profile consists of a collection of 3D geometric primitives that form a tessellation of a volume, i.e. a partitioning of the space. Partitioned Space refers to data which is equivalent to Full Topology with the added requirement that all “blank” space is filled. This becomes important when dealing with very detailed building interiors, geological data, or bathymetric data.

6.13.2 Class diagrams

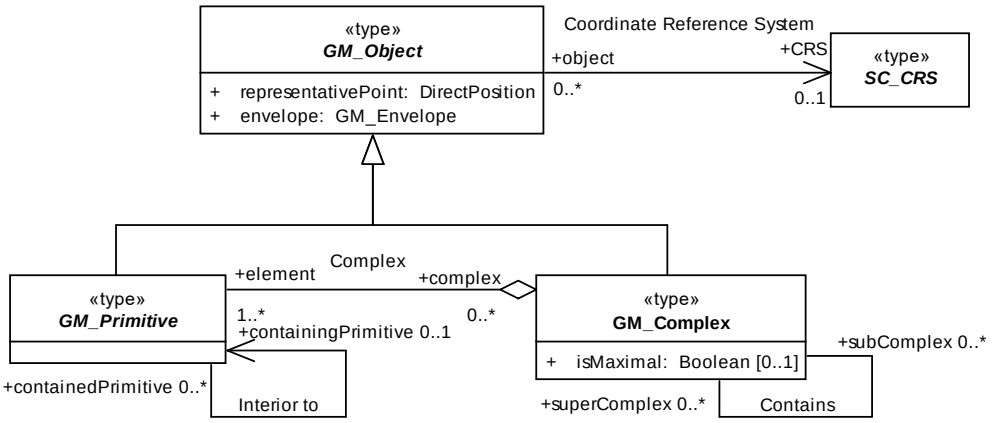


Figure 6.13.1 – Geometry object

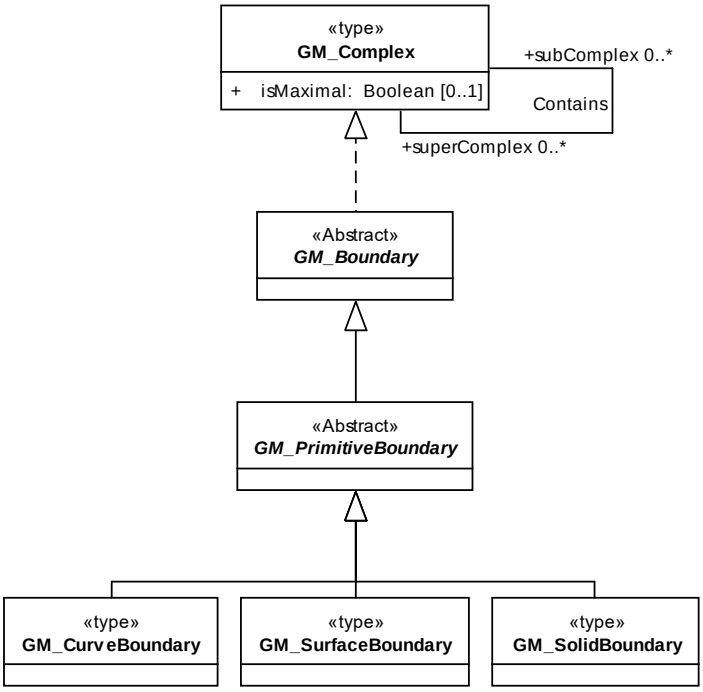


Figure 6.13.2 – Geometry boundary

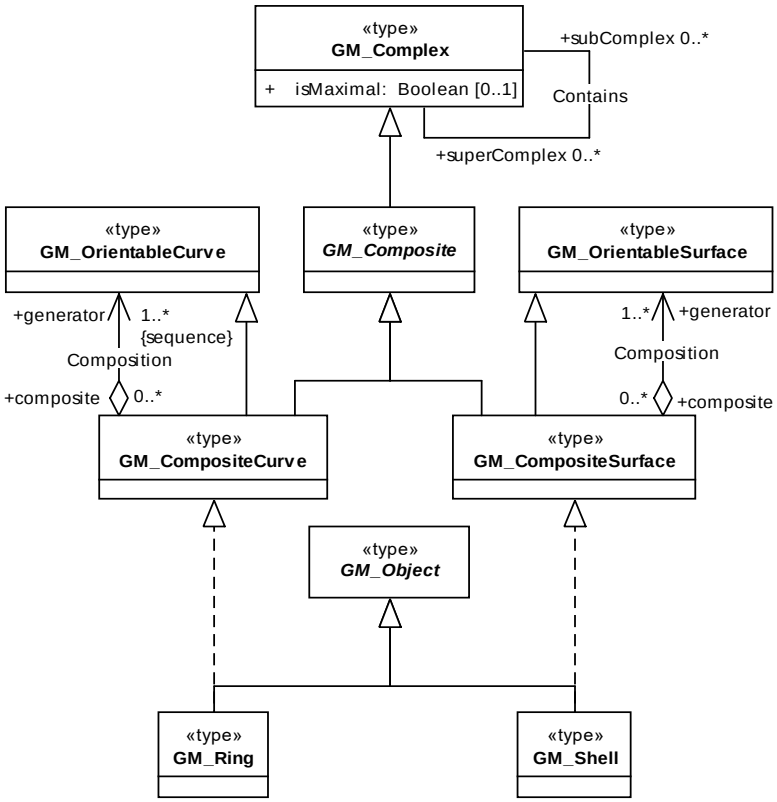


Figure 6.13.3 – Geometry boundary components

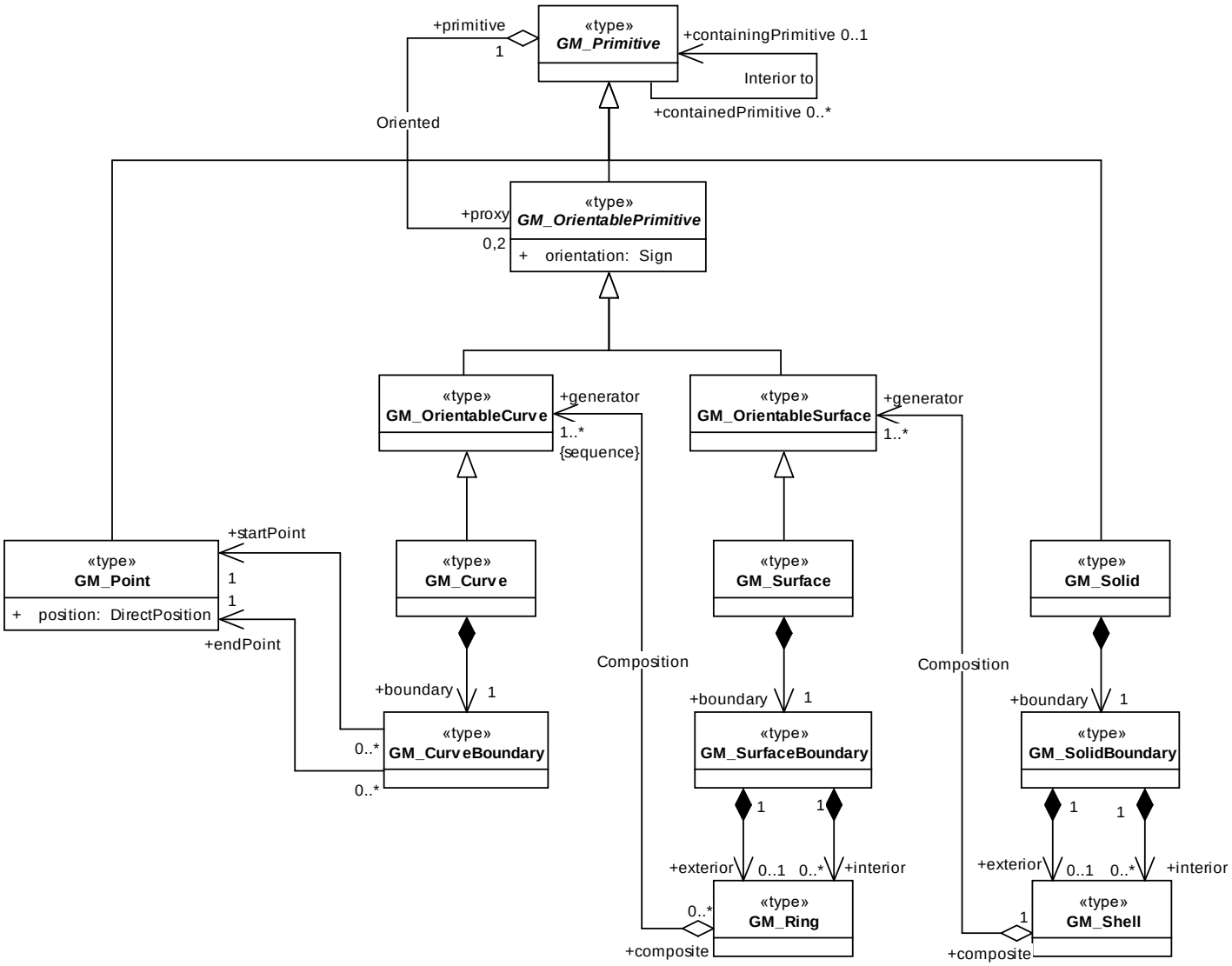


Figure 6.13.4 – Geometry primitive

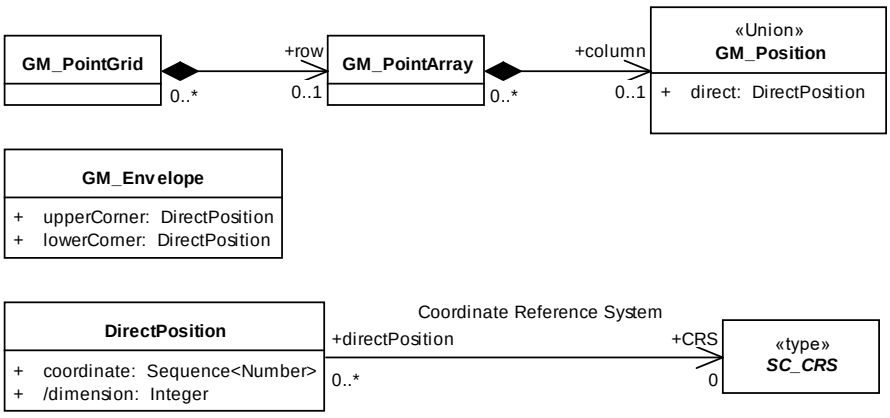


Figure 6.13.5 – Coordinate package

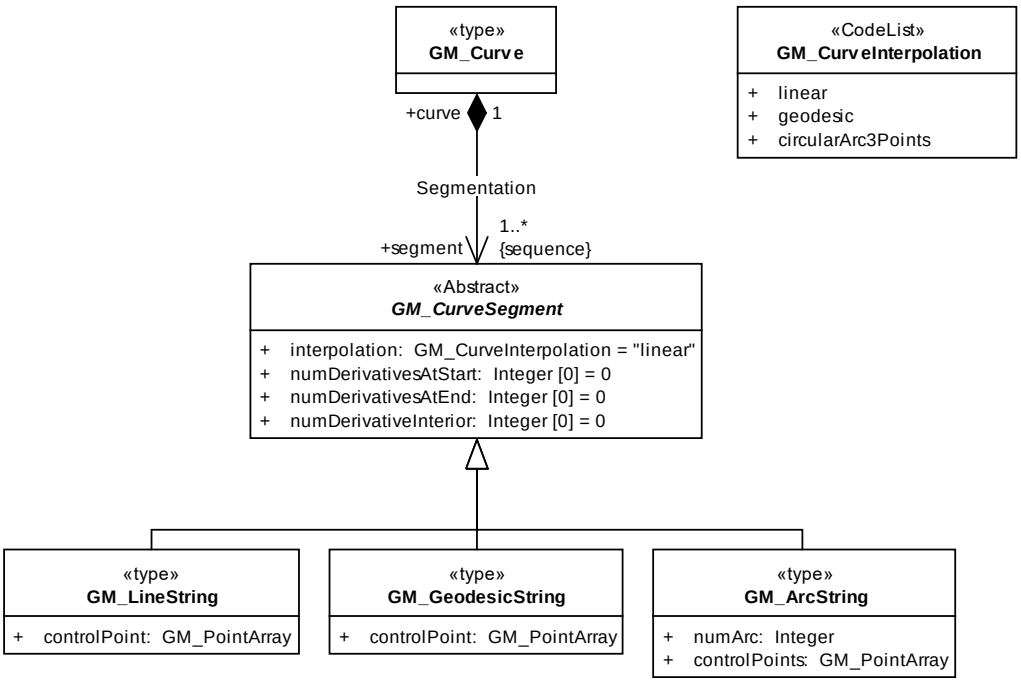


Figure 6.13.6 – Curve segment

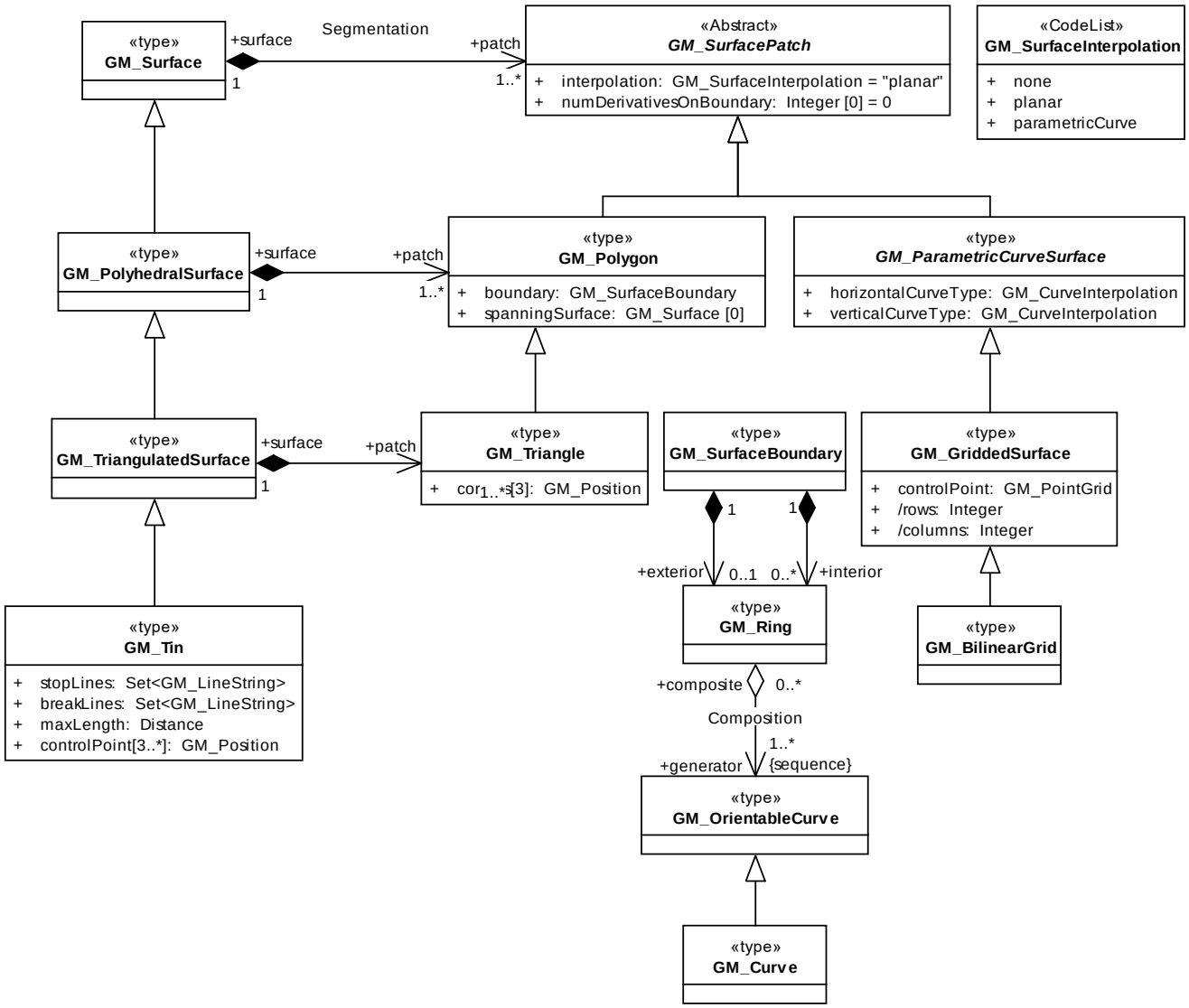


Figure 6.13.7 – Surface patch

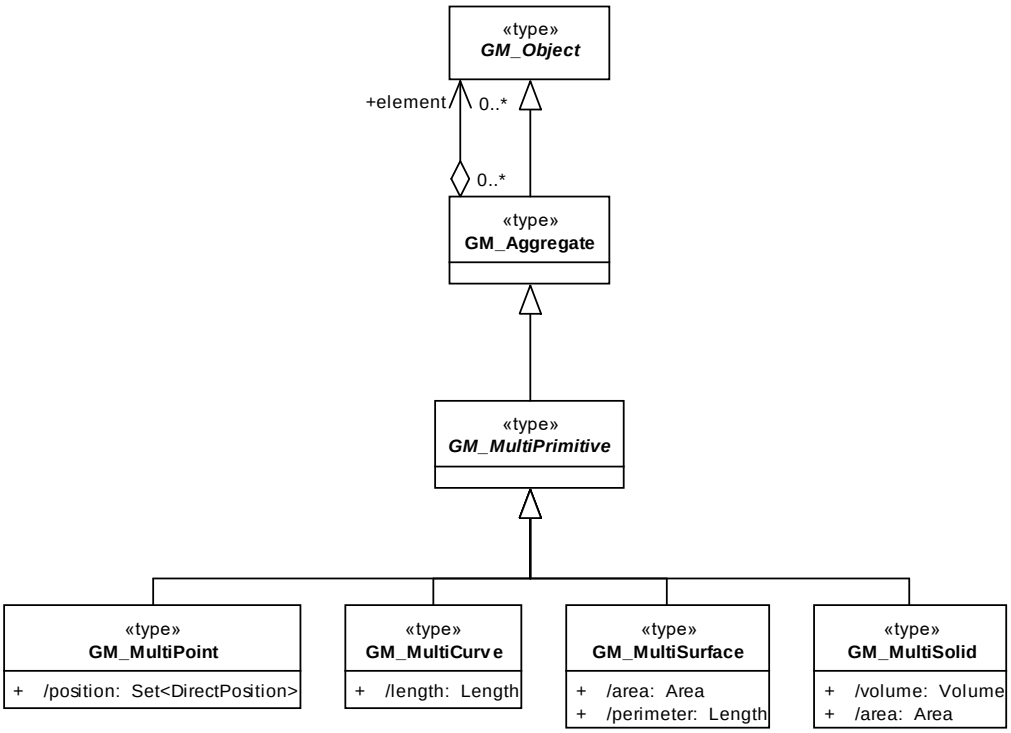


Figure 6.13.8 – Geometry aggregation

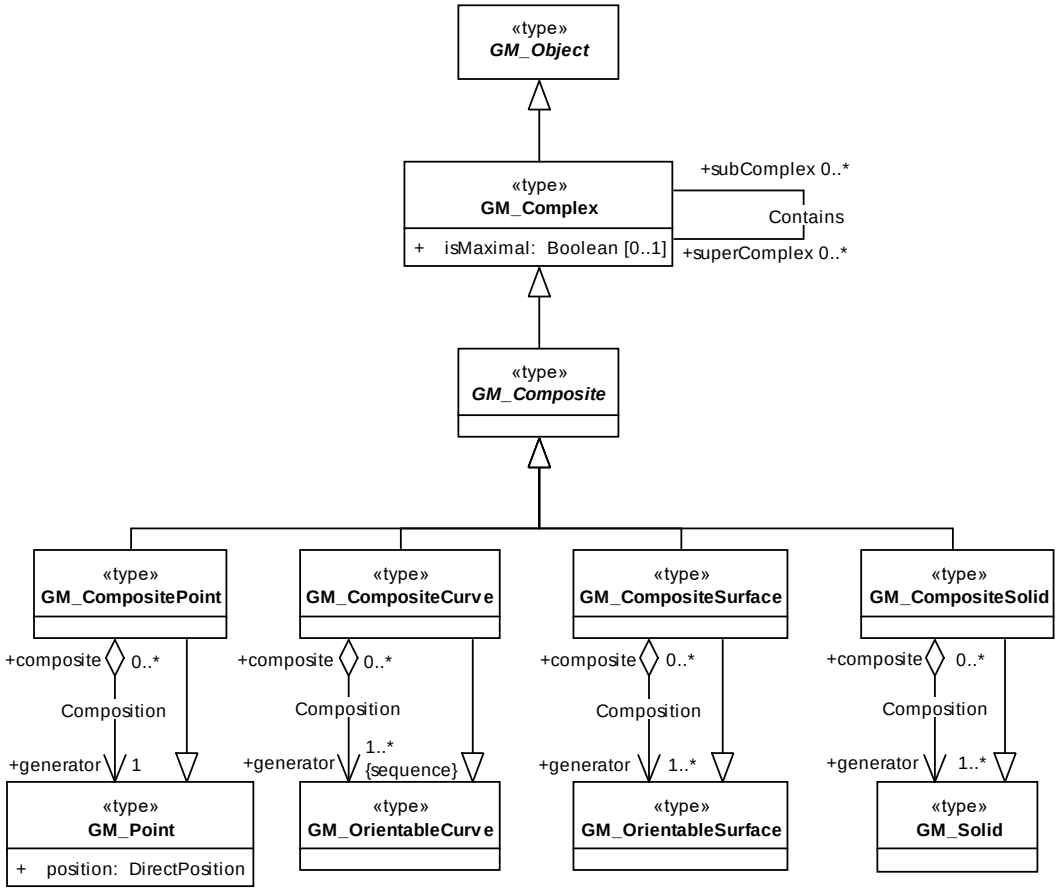


Figure 6.13.9 – Geometry composites

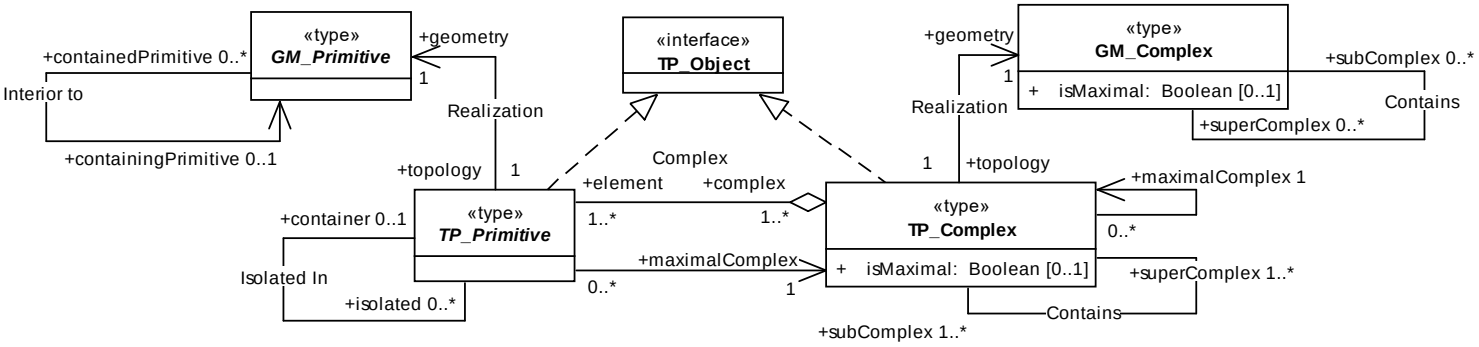


Figure 6.13.10– Topology object

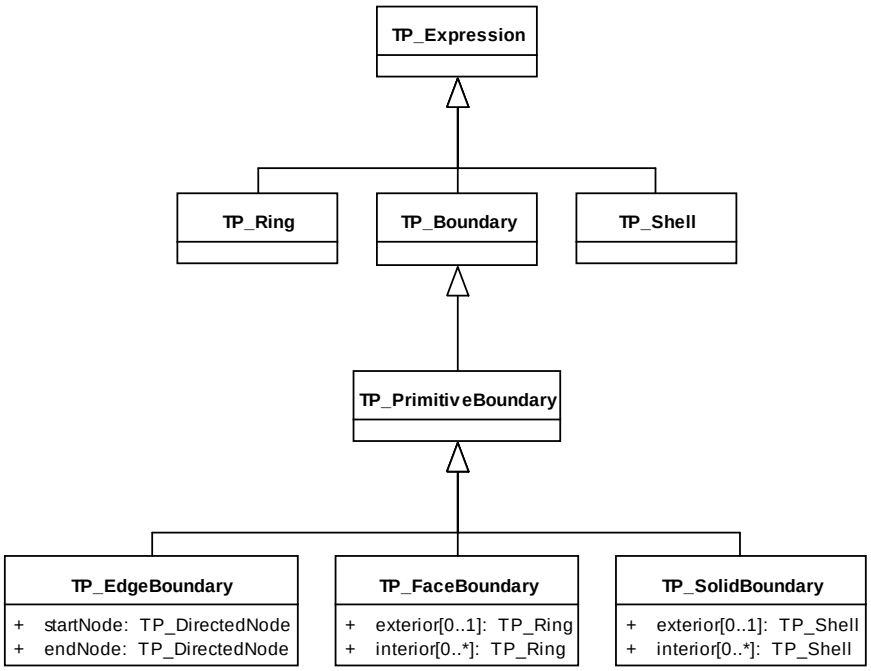


Figure 6.13.11 – Topology boundary

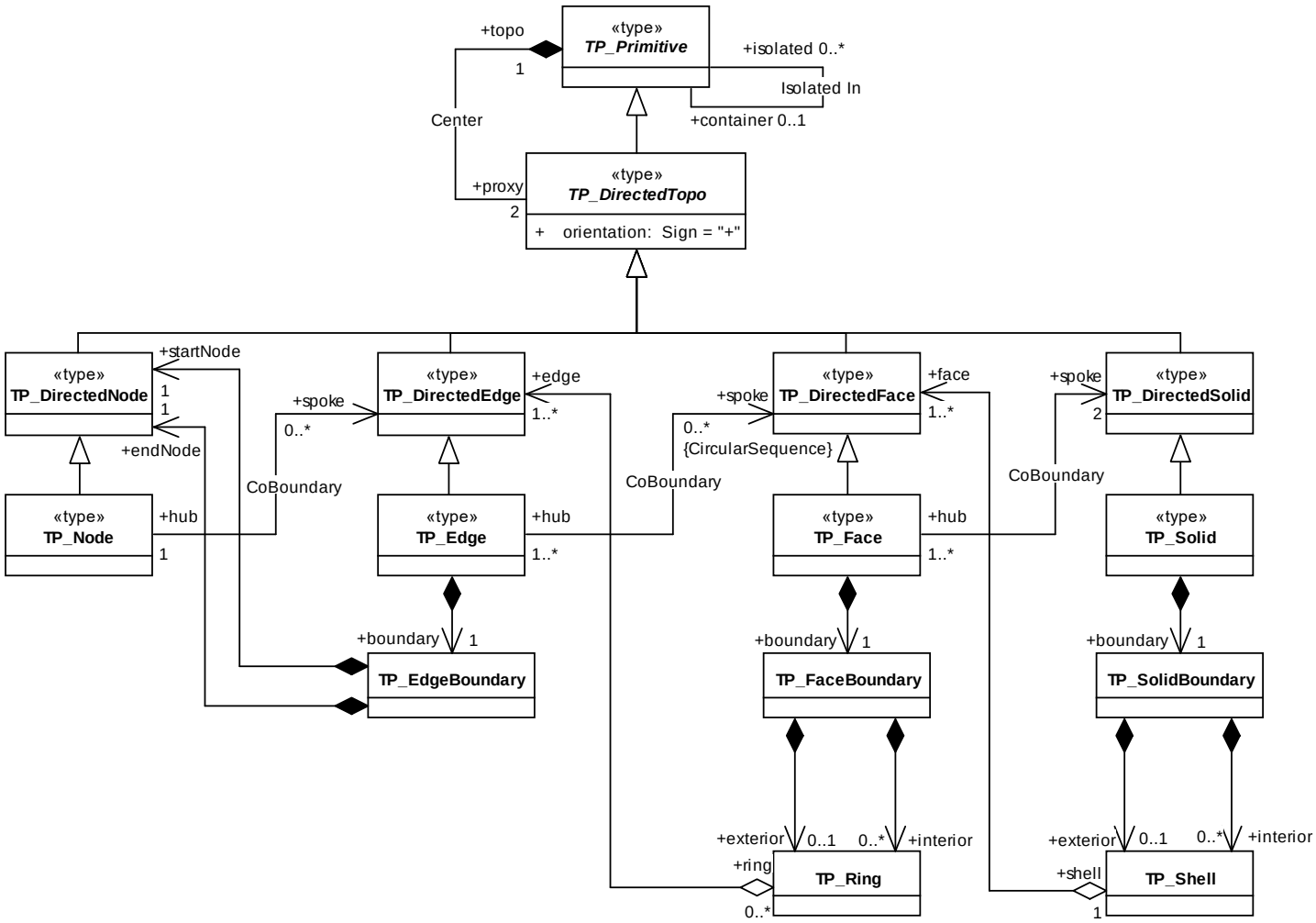


Figure 6.13.12 – Topology primitive

6.13.3 Included constructs

The following is a list of all the included classes preceded by their ISO 19107 section number. If a class or one of its associations is specialized in this profile the section in this profile defining the specialization is also referenced.

- 6.2.2 GM_Object – specializations 6.13.4.1(S1), 6.13.4.2(S2)
- 6.3.2 GM_Boundary
- 6.3.4 GM_PrimitiveBoundary
- 6.3.5 GM_CurveBoundary
- 6.3.6 GM_Ring
- 6.3.7 GM_SurfaceBoundary
- 6.3.8 GM_Shell
- 6.3.9 GM_SolidBoundary
- 6.3.10 GM_Primitive – specializations 6.13.4.3(S5), 6.13.4.4(S7), 6.13.4.19(S26)
- 6.3.11 GM_Point
- 6.3.13 GM_OrientablePrimitive
- 6.3.14 GM_OrientableCurve – specialization 6.13.4.5(S11)
- 6.3.15 GM_OrientableSurface – specialization 6.13.4.6(S12)
- 6.3.16 GM_Curve – specialization 6.13.4.5(S11)

- 6.3.17 GM_Surface – specialization 6.13.4.6(S12), 6.13.4.8(S15)
- 6.3.18 GM_Solid – specialization 6.13.4.9(S16)
- 6.4.1 DirectPosition – specialization 6.13.4.10(S17)
- 6.4.3 GM_Envelope
- 6.4.5 GM_Position – specialization 6.13.4.11(S18)
- 6.4.6 GM_PointArray
- 6.4.6 GM_PointGrid
- 6.4.8 GM_CurveInterpolation – specialization 6.13.4.12(S19)
- 6.4.9 GM_CurveSegment – specializations 0(S13), 6.13.4.13(S20)
- 6.4.10 GM_LineString
- 6.4.12 GM_GeodesicString
- 6.4.14 GM_ArcString
- 6.4.32 GM_SurfaceInterpolation – specialization 6.13.4.14(S21)
- 6.4.34 GM_SurfacePatch – specializations 6.13.4.15(S22), 6.13.4.16(S23)
- 6.4.35 GM_PolyhedralSurface
- 6.4.36 GM_Polygon – specialization 6.13.4.17(S24)
- 6.4.37 GM_TriangulatedSurface
- 6.4.38 GM_Triangle
- 6.4.39 GM_Tin
- 6.4.40 GM_ParametricCurveSurface
- 6.4.41 GM_GriddedSurface
- 6.4.45 GM_BilinearGrid
- 6.5.2 GM_Aggregate
- 6.5.3 GM_MultiPrimitive
- 6.5.4 GM_MultiPoint
- 6.5.5 GM_MultiCurve
- 6.5.6 GM_MultiSurface
- 6.5.7 GM_MultiSolid
- 6.6.2 GM_Complex – specializations 6.13.4.18(S25), 6.13.4.24(S34)
- 6.6.3 GM_Composite
- 6.6.4 GM_CompositePoint
- 6.6.5 GM_CompositeCurve
- 6.6.6 GM_CompositeSurface
- 6.6.7 GM_CompositeSolid
- 7.2.2 TP_Object
- 7.3.2 TP_Boundary
- 7.3.4 TP_PrimitiveBoundary
- 7.3.5 TP_EdgeBoundary
- 7.3.6 TP_FaceBoundary
- 7.3.7 TP_SolidBoundary
- 7.3.8 TP_Ring
- 7.3.9 TP_Shell
- 7.3.10 TP_Primitive – specializations 6.13.4.19(S26), 6.13.4.20(S27)
- 7.3.11 TP_DirectedTopo

- 7.3.12 TP_Node
- 7.3.13 TP_DirectedNode
- 7.3.14 TP_Edge
- 7.3.15 TP_DirectedEdge
- 7.3.16 TP_Face – specialization 6.13.4.21(S31)
- 7.3.17 TP_DirectedFace
- 7.3.18 TP_Solid
- 7.3.19 TP_DirectedSolid
- 7.3.20 TP_Expression
- 7.4.2 TP_Complex – specializations 6.13.4.22(S32), 6.13.4.23(S33), 6.13.4.24(S34)

6.13.4 Specializations

6.13.4.1 GM_Object (S1)

The representativePoint operation of GM_Object is changed to an optional attribute.

```
GM_Object::representativePoint[0,1] : DirectPosition
```

6.13.4.2 GM_Object (S2)

The envelope operation of GM_Object is changed to an optional attribute.

```
GM_Primitive::envelope[0,1] : GM_Envelope
```

6.13.4.3 GM_Primitive (S5)

All GM_Primitives are disjoint.

```
context GM_Primitive inv:
forAll( p1, p2 | not p1.intersects( p2 ) )
```

6.13.4.4 GM_Primitive (S7)

The InteriorTo association of GM_Primitives is used like the TP_Primitive::IsolatedIn association. The InteriorTo association only associates GM_Primitives with a dimensional difference greater than one. The multiplicity of the superElement association role is constrained from [0..n] to [0..1].

```
GM_Primitive::containingPrimitive[0..1] : GM_Primitive                    ([0..n])

context GM_Primitive inv:
containedPrimitive->forAll( p | p.dimension() < self.dimension() - 1)
```

6.13.4.5 GM_OrientableCurve, GM_Curve (S11)

The boundary operation of GM_OrientableCurve is changed in GM_Curve to a composition relationship to GM_CurveBoundary.

```
GM_Curve::boundary[1] : GM_CurveBoundary
```

6.13.4.6 GM_OrientableSurface, GM_Surface (S12)

The boundary operation of GM_OrientableSurface is changed in GM_Surface to a composition relationship to GM_SurfaceBoundary.

```
GM_Surface::boundary[1] : GM_SurfaceBoundary
```

6.13.4.7 GM_CurveSegment (S13)

The multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained from [0,1] to [1].

```
GM_CurveSegment::curve[1] : GM_Curve                                        ([0,1])
```

6.13.4.8 GM_Surface (S15)

The interior of a GM_Surface patch is described by GM_SurfacePatches. The multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained from [0..n] to [1..n].

```
GM_Surface::patch[1..n] : GM_SurfacePatch                    ([0..n])
```

6.13.4.9 GM_Solid (S16)

The boundary operation of GM_Solid is changed to a composition relationship to GM_SolidBoundary.

```
GM_Solid::boundary[1] : GM_SolidBoundary
```

6.13.4.10 DirectPosition (S17)

The multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained from [0,1] to [0]. This forces all positions of a primitive to use the same coordinate reference systems.

```
DirectPosition::coordinateReferenceSystem[0] : SC_CRS        ([0,1])
```

6.13.4.11 GM_Position (S18)

The union GM_Position always uses the direct variant. The indirect variant is not used.

```
GM_Position::direct[1] : DirectPosition                      ([0,1])
GM_Position::indirect[0] : GM_PointRef                       ([0,1])
```

6.13.4.12 GM_CurveInterpolation (S19)

The GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".

```
GM_CurveInterpolation::
linear
geodesic
circularArc3Points
```

6.13.4.13 GM_CurveSegment (S20)

The multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained from [0,1] to [0].

```
GM_CurveSegment::numDerivativesAtStart[0] : Integer          ([0,1])
GM_CurveSegment::numDerivativesInterior[0] : Integer         ([0,1])
GM_CurveSegment::numDerivativesAtEnd[0] : Integer            ([0,1])
```

6.13.4.14 GM_SurfaceInterpolation (S21)

The GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.

```
GM_SurfaceInterpolation::
none
planar
parametricCurve
```

6.13.4.15 GM_SurfacePatch (S22)

The multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained from [0,1] to [1].

```
GM_SurfacePatch::surface[1] : GM_Surface                     ([0,1])
```

6.13.4.16 GM_SurfacePatch (S23)

The multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained from [0,1] to [0].

```
GM_SurfacePatch::numDerivativesOnBoundary[0] : Integer       ([0,1])
```

6.13.4.17 GM_Polygon (S24)

The multiplicity of the spanningSurface attribute of GM_Polygon is constrained from [0..1] to [0].

```
GM_Polygon::spanningSurface[0] : GM_Surface                    ([0,1])
```

6.13.4.18 GM_Complex (S25)

The isMaximal operation of GM_Complex is changed to an optional Boolean attribute.

```
GM_Complex::isMaximal[0,1] : Boolean
```

6.13.4.19 GM_Primitive, TP_Primitive (S26)

The multiplicity of the geometry role of the Realization association between TP_Primitive and GM_Primitive is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Primitive and TP_Primitive is constrained from [0..n] to [1].

```
TP_Primitive::geometry[1] : GM_Primitive                       ([0,1])
GM_Primitive::topology[1] : TP_Primitive                      ([0..n])
```

6.13.4.20 TP_Primitive (S27)

The association variant of the maximalComplex property of TP_Primitive is used.

6.13.4.21 TP_Face (S31)

The multiplicity of the spoke role of the CoBoundary association between TP_Face and TP_DirectedSolid is constrained from [0..2] to [2].

```
TP_Face::spoke[2] : TP_DirectedSolid                           ([0..2])
```

6.13.4.22 TP_Complex (S32)

The association variant of the maximalComplex property of TP_Complex is used.

6.13.4.23 TP_Complex (S33)

The isMaximal operation of TP_Complex is changed to an optional Boolean attribute.

```
TP_Complex::isMaximal[0,1] : Boolean
```

6.13.4.24 GM_Complex, TP_Complex (S34)

The multiplicity of the geometry role of the Realization association between TP_Complex and GM_Complex is constrained from [0,1] to [1]. The multiplicity of the topology role of the Realization association between GM_Complex and TP_Complex is constrained from [0,1] to [1].

```
TP_Complex::geometry[1] : GM_Complex                           ([0,1])
GM_Complex::topology[1] : TP_Complex                           ([0,1])
```

7 Use cases

Paper maps are a two-dimensional representation of geographic information. Elevation data is added to this two-dimensional representation by means of isohypses – contour lines. Traditionally digital representation of geographic information has taken a similar approach to this, where the information represented has been essentially two-dimensional with elevation data functioning as additional attribution added to horizontal positions. The representation remained in essence two-dimensional. With the increase in both computing and storage capacity it has become possible and also desirable to model geographic information more accurately and to be able to model aspects of geographic information that are inherently three-dimensional. Applications that benefit from three-dimensional representation include: urban environment, transportation network, geological, and meteorological modelling.

The following use cases offer an overview of possible implementations. These use cases should help users to identify the appropriate geometry and topology profile.

UC#	Use Case	Initiating Actor	Receiving Actor	Minimum Required UML Profiles[1]	Description
1	**Visualize Geospatial Information**				
1.1	Visualize two dimensional information (2D)	General Operational User	Mission Planner	L1.2D.2d	Traditional “map” view of digital data. No topology requirements. NOTE: Possible feature overlapping should be taken care of by portrayal finishing rules.
1.2	Visualize three dimensional information (3D)	General Operational User	Mission Planner	L1.3D.3d	3 Dimensional “map” view of digital data. No topology requirements.
2	**Processing and Analysis of Geospatial Data**				
2.1	Provide clean (quality) 2D or 2½D for lines	Geospatial Unit	Geospatial Specialist	L3.2D.1d	No specific requirement for encoded topology. Data must be “clean” (e.g. no self intersecting lines, features must break at intersections, overshoots, undershoots)
2.2	Provide clean (quality) 2D or 2½D data for lines and areas	Geospatial Unit	Geospatial Specialist	L3.2D.2d	No specific requirement for encoded topology. Data must be “clean” (e.g. no self intersecting lines, features must break at intersections, no gaps, overshoots, undershoots)
2.3	Provide clean (quality) 3D data for lines, areas and solids	Geospatial Unit	Geospatial Specialist	L3.3D.3d	No specific requirement for encoded topology. Data must be “clean” (e.g. no self intersecting lines, features must break at intersections, features must connect at the same Z coordinate location, no gaps, overshoots, undershoots)

[1] See Table 8.3 to identify the corresponding GML profile.

UC#	Use Case	Initiating Actor	Receiving Actor	Minimum Required UML Profiles[1]	Description
2.4	Provide clean (quality) 2D or 2½D data for lines and areas	Geospatial Unit	Geospatial Specialist	L4.2D.2d	Topology to be built from geometry as part of subsequent data processing. No specific requirement for encoded topology. Data must be "clean" (e.g. no self intersecting lines, features must break at intersections, no gaps, overshoots, undershoots)
2.5	Provide clean (quality) 3D data for lines, areas and solids	Geospatial Unit	Geospatial Specialist	L4.3D.3d	Topology to be built from geometry as part of subsequent data processing. No specific requirement for encoded topology. Data must be "clean" (e.g. no self intersecting lines, features must break at intersections, features must connect at the same Z coordinate location, no gaps, overshoots, undershoots)
2.6	Proximity Analysis	Geospatial Unit	Geospatial Specialist	L1.2D.2d	Allows the user to find a feature or feature category within a certain distance of other features. This type of analysis is useful in determining Helicopter Landing Zones, Drop Zones, Bivouac Sites and Main Supply Routes. Coverages would be dependent upon the type of analysis.
2.7	Helicopter Landing Zones	Geospatial Unit	Geospatial Specialist	L1.3D.3d	Used to locate acceptable areas for helicopter landings. Coverage(s) used: terrain, vegetation.
3	**Network Analysis - Routing**				
3.1	Calculate the shortest path 2D and 2½D data	Geospatial Specialist	Geospatial Specialist	L5.2D.1d	No specific requirement for encoded topology. Topology to be built from geometry as part of the data processing. Data must be "clean" (e.g. no self intersecting lines, features must break at intersections, no gaps, overshoots, undershoots)
3.2	Calculate the shortest path with 3D data	Geospatial Specialist	Geospatial Specialist	L5.3D.1d	No specific requirement for encoded topology. Topology to be built from geometry as part of the data processing. Data must be "clean" (e.g. no self intersecting lines, features must break at intersections, no gaps, overshoots, undershoots). Note: When dealing with 2½D data overpass/underpass intersections are not reflected as real world instances as the Z Value must be the same for the common coordinate position.
3.3	Time Contours	Geospatial Specialist	Geospatial Specialist	L5.2D.2d L5.3D.3d	Shows the time to reach certain points by displaying lines of equal time. Coverage(s) used: surface materials, terrain, vegetation, transportation.

UC#	Use Case	Initiating Actor	Receiving Actor	Minimum Required UML Profiles[1]	Description
3.4	Calculate the least cost path in a network	Geospatial Specialist	Geospatial Specialist	L5.2D.2d L5.3D.2d	This level of network analysis requires the ability to add external variables into the decision making process to determine least cost path. In other words the most direct route may not be the preferred route based on some extenuating circumstances (e.g. adjoining river flooding, bridge destroyed, avoidance areas, etc.)
3.5	Calculate the least cost path – cross country movement	Geospatial Specialist	Geospatial Specialist	L6.2D.2d L6.3D.2d	While not technically network analysis this use case represents route analysis which requires the ability to determine the best route based on topologically connected soil type boundaries and terrain information or slope polygons. This use case also requires fully partitioned space meaning no areas are allowed to be undefined or "whitespace". The least cost path would reflect those areas capable of supporting the vehicle while at the same time avoiding steep slopes. Coverage(s) used: surface materials, surface configuration, vegetation, transportation
3.6	Routes of flight (Air Traffic Service ATS)	Geospatial Specialist	Geospatial Specialist	L6.3D.3d	This use case is provided to describe an example for navigation in a 3D space. Locations can be modelled as points, lines, areas and solids. L6 does not allow for any undefined space.
3.7	Instrument flight procedure analysis	Terminal instrument procedure designer	Aircraft flight crew member	L2.3D.2d	Instrument flight procedure analysis uses vertical obstruction datasets combined with aerodrome runway data to determine the safe instrument procedure location and altitude for safe approach to the runway.
3.8	Maritime use case	Maritime Specialist	Maritime Specialist	L6.3D.3d	This use case is provided to describe an example for navigation in a 3D space. Locations can be modelled as points, lines, areas and solids. L6 does not allow for any undefined space. (Littoral) The definition of territorial waters requires an offset from the coastline.
3.9	Submarine use case	Maritime Specialist	Maritime Specialist	L6.3D.3d	This use case is provided to describe an example for navigation in a 3D space. Locations can be modelled as points, lines, areas and solids (e.g. under water obstacles such as pinnacles, wrecks). L6 does not allow for any undefined space.
3.10	Sonar propagation	Maritime Specialist	Maritime Specialist	L3.3D.3d	This use case integrates bathymetric and oceanographic information to determine areas for the optimum use of the sonar.
4	**Terrain Analysis**				

UC#	Use Case	Initiating Actor	Receiving Actor	Minimum Required UML Profiles[1]	Description
4.1	Calculate line-of-sight	General Operational User	Geospatial Specialist	L3.3D.3d	This use case is provided to describe a simple ability to determine the line of sight between any two points. No specific topology requirement exists, however some algorithms might be faster when using explicit topology. Note 1: 2 ½ D terrain modelling is implicit to the 2D spatial schema profile. Note 2: Obstacles (e.g. buildings) in the line of sight might be modelled as 3D solids.
4.2	Calculate area-of-sight; Generate surface models	General Operational User	Geospatial Specialist	L3.3D.3d	The area of sight (view shed) use case is a generalization of the line of sight use case where all points visible from a single location are calculated. Surface features may be regarded as obstacles in certain types of terrain analysis. Topological relationships may exist between the cultural features in order to simplify the process of determining view obstructed areas. See 4.1 Notes.
4.3	Aerial Detection	General Operational User	Geospatial Specialist	L1.3D.3d	Indicates distances at which incoming targets become visible to an observer on the ground. Coverage(s) used: surface configuration, vegetation.
4.4	Cover and Concealment	General Operational User	Geospatial Specialist	L1.2D.2d	Predicts the probability of detection of a ground target by visual aerial surveillance based on vegetation type and canopy closure. Coverage(s) used: vegetation.
4.5	Gap Crossing	Geospatial Specialist	Geospatial Specialist	L1.2D.2d	Computes the span and swim capability for user-specified vehicles. Coverage(s) used: surface drainage
5	**Common Operational Picture; Recognized Environmental Picture; Data Fusion**				This use case defines a more complex form of fusion. Multi-intelligence data fusion becomes increasingly feasible in near real time, the integration of accurate, comprehensive and up-to-date geospatial data within this fusion process becomes vital. This geospatial data forms the fundamental base layer for the Joint Intelligence Picture and thus it is important that it conforms to well-defined standards. To form the Recognised Environmental Picture (REP) within the Joint Operational Picture (JOP) fusion of geographic, hydrographic, oceanographic, aeronautical and meteorological data will be necessary. The use of web services can facilitate the fusion, however topological and geometric harmonization might be required. Topological information is essential for geometric fusion/conflation and is used for quality assurance. For meteorological data time becomes an issue.

UC#	Use Case	Initiating Actor	Receiving Actor	Minimum Required UML Profiles[1]	Description
5.1	Support to civilian emergency management	Geospatial Specialist	Geospatial Specialist	L3.2D.2d	Military support to civilian agencies in times of emergencies and crisis is becoming more common place. This use case attempts to define the requirements for data, e.g. meteorological, building information, traditional maps, imagery, elevation data, …
5.2	Harbour Security	Maritime Specialist	Maritime Specialist	L5.3D.3d	Port security involves the integration of hydrographic and topographic data.
5.3	Facility management / Security	Geospatial Specialist	Geospatial Specialist	L5.3D.3d	This use case describes the requirements for managing facility operations and security. It requires detailed knowledge of the relationships between features that compose a given facility. Topology information is crucial for analyzing these relationships.
5.4	Fuse data to produce a REP, JOP or COP	Geospatial Specialist	Geospatial Specialist	L1.2D.2d (data / objective dependent)	This use case defines a more complex form of fusion. Multi-intelligence data fusion becomes increasingly feasible in near real time, the integration of accurate, comprehensive and up-to-date geospatial data within this fusion process becomes vital. This geospatial data forms the fundamental base layer for the Joint Intelligence Picture and thus it is important that it conforms to well-defined standards. To form the Recognised Environmental Picture (REP) within the Joint Operational Picture (JOP) fusion of geographic, hydrographic, oceanographic, aeronautical and meteorological data will be necessary. Standard tools are required to allow this process to be achieved seamlessly despite differing data standards across the international geospatial community.
6	**Tracking and Navigation**				Capability to identify any objects location, tracking its movement and estimating where a moving object will be at a given time.
6.1	Positioning	Operational User	Operational User	L1.2D.2d	For example GPS location and navigation such as friendly force tracking / blue force tracking. Geometry requirements for this use case may only include point and line. In cases where uncertainty of position or location exists, topology may be required. It is anticipated that this use case will require temporal information as well.

UC#	Use Case	Initiating Actor	Receiving Actor	Minimum Required UML Profiles[1]	Description
6.2	Localization	Operational User	Operational User	L1.2D.2d	Identifying the location of hostile forces, this could include sub-surface, surface and airborne surveillance. Geometry requirements for this use case may only include point and line. In cases where uncertainty of position or location exists, topology may be required. It is anticipated that this use case will require temporal information as well.
6.3	Estimating position	Operational User	Operational User	L1.2D.2d	Estimation where a moving object will be at a given time. Geometry requirements for this use case may only include point and line. This use case will require temporal information. In cases where uncertainty of position or location exists, topology may be required to calculate a feasible region.
6.4	Generate datasets and imagery sets for targeting support	Geospatial Specialist	Geospatial Specialist	L3.3D.3d	The use of precision weapons requires a comprehensive targeting capability, with target analysis and co-ordinate calculation being based on rigorously applied procedures. A fundamental element of this process is the provision of accurate three dimensional geospatial data and imagery to well defined standards within a predetermined worldwide geodetic framework. Topological higher level data might be required, dependent on the intended use.
7	**Targeting Support**				
7.1	Generate datasets and imagery sets for targeting support	Geospatial Specialist	Geospatial Specialist	L3.3D.3d	The use of **precision weapons requires a comprehensive targeting capability**, with target analysis and co-ordinate calculation being based on rigorously applied procedures. A fundamental element of this process is the provision of accurate **three dimensional geospatial data** and imagery to well defined standards within a predetermined worldwide geodetic framework.

NOTE: In most cases where L3 input data is used, L1 data would be usable, but would require more computational effort.

8 GML realization of the DGIWG Profiles of ISO 19107

8.1 Introduction to the Geography Markup Language (GML)

GML enables an interoperable transfer of geo-data between distributed systems. It offers an open, vendor-neutral framework for the definition of application schemas that describe the geometry and characteristics of real world objects in a standardised way. GML can be used for storage and transport of geographic data alike. Additionally, a number of OGC web-services use GML for data exchange, especially for encoding requests and responses of Web Feature Services (WFS). GML offers the capabilities to enable an Internet-based infrastructure of interconnected geographic information sources and processing services. An approach that is used in local, national and international spatial data infrastructures alike.

The GML standard provides a set of rules and constructs for encoding geospatial and non-geospatial information, which can be adapted to the specific needs of a user domain. These domain specific needs then define a tailored GML application schema which defines that Community of Interests associated exchange format. These schema files define the elements and object types that can be used in a GML file and determine how the data in a GML file has to be structured (for example which elements, tags or hierarchy could be expected). This makes it possible to validate GML files against their application schema and verify if they are well formed (syntactically) and valid. Figure 8.1 shows the connections between the GML standard, application schemas and data instances.

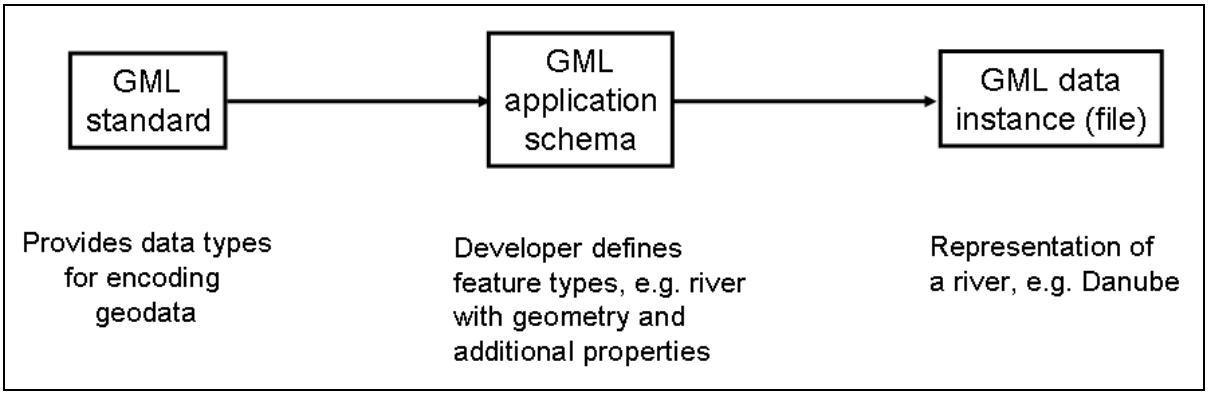


Figure 8.1 – Relationship between GML standard, application schema and data instance

GML enables the encoding of a number of phenomena for example the encoding of dynamic features, spatial and temporal topology, complex geometric property types and coverages. Because it is rather unlikely that there is an application that needs all constructs offered by the GML standard, it is possible to reduce the number of supported elements and types to that required by a particular application. Therefore GML offers the possibility of subsetting the specification by generating profiles.

Profiles of GML are generated by selecting and restricting the constructs offered by the GML standard, which are necessary to accomplish specific functions in real application systems. One GML profile could be the common basis for different application schemas of one or more user domain(s). All application schemas that conform to a specific profile have to use the same restricted capabilities. Therefore a profile simplifies software development, increases performance and strengthens interoperability.

To define a specific GML profile, it is necessary to identify the requirements of the specific user domain(s). The GML elements and types needed for data representation have to be determined. Annex F of the GML standard provides guidelines and XSLT-scripts for sub-setting the GML schemata defined in the standard.

The following Figure 8.2 shows the relationship between GML standard, profile and application schema. According to ISO 19136:2007 the building of an application schema is a two-part process. The profile restricts the types and elements consistent with the complete GML standard. The application schema then uses these types to define a domain specific model by extension or inclusion. In this respect extension can be understood as defining the representation of real world objects on top of the provided GML constructs.

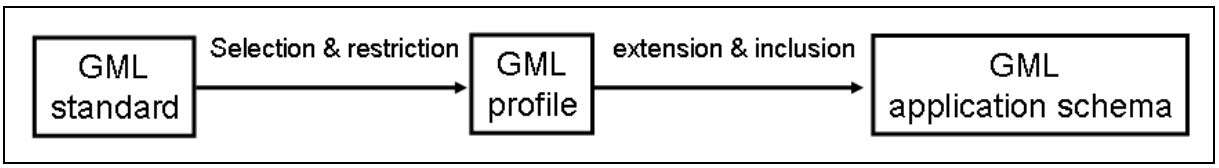


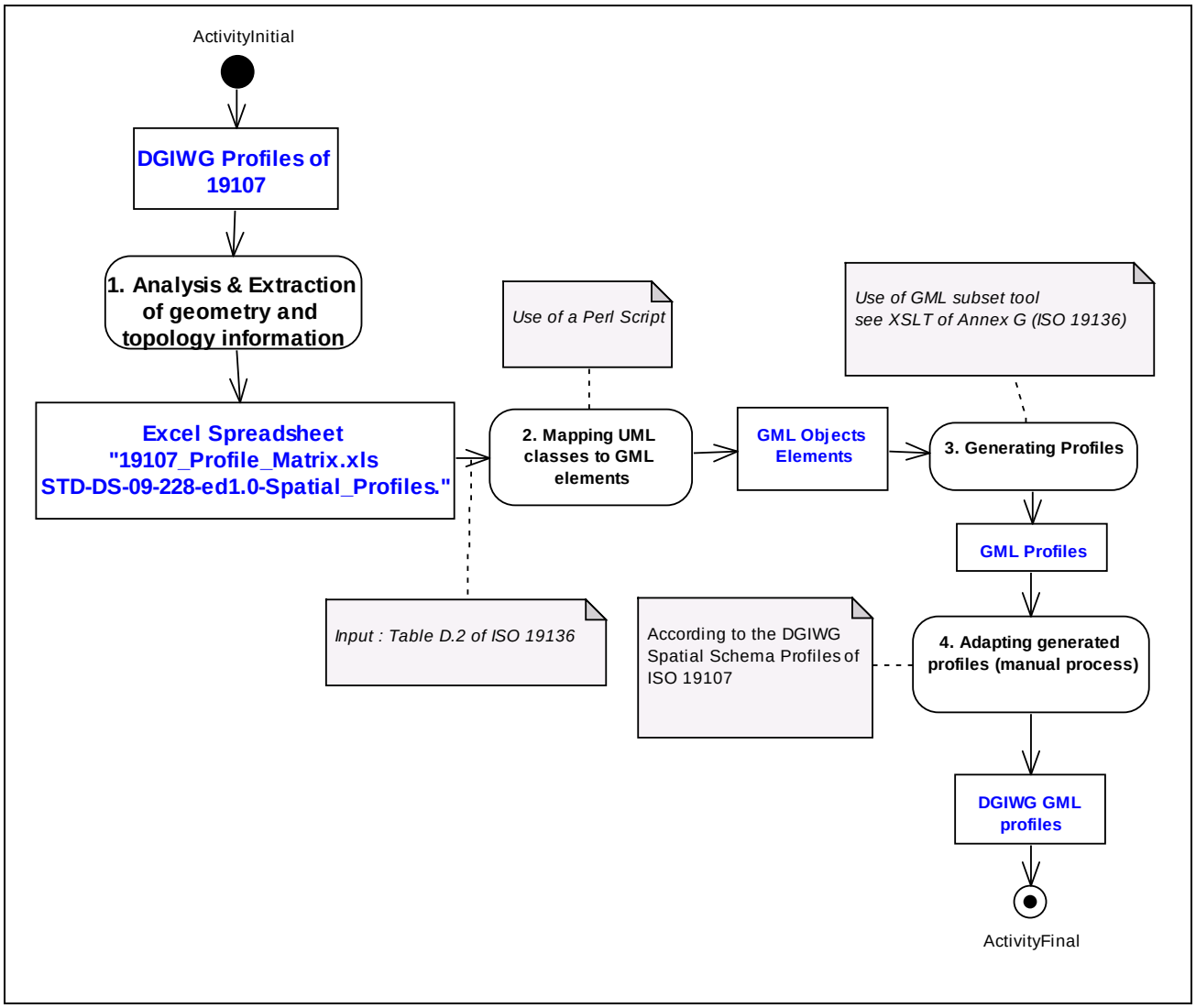
Figure 8.2 – Relationship between GML standard, profile and application schema

Note: A GML application schema shall reference the full GML schema in the schemaLocation attribute of the <import> element. A GML application schema document conforming to one or more GML Profiles shall provide an appInfo annotation element <gml:gmlProfileSchema> for every profile in the root schema document <schema> element where the value is a schema location of the profile schema. Note that an application schema may conform to multiple profiles. [ISO 19136:2007]

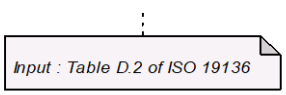
8.2 Steps for generating the DGIWG GML profiles for the DGIWG Spatial Schema Profiles of ISO 19107

The DGIWG Profiles of ISO 19107 define several profiles for geometry and topology. For each of these profiles the UML classes are defined in a profile matrix which can be found in the attachment to the standard. Figure 8.3 shows the steps described below.

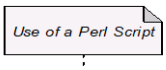
1. Mapping of the extracted UML classes to their equivalent GML object elements
2. Generating the different profiles with the GML subset tool provided by the GML specification taking the identified GML object elements as parameters
3. Manually adapt the generated profiles according to the specializations defined by the DGIWG Profiles of ISO 19107



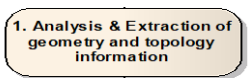
Legend :



Comment on a arrow to specify necessary input



Comment on an activity (use of a script, tool)



An activity / Step of the generating process

Result Object / Element after a step

Figure 8.3 – Workflow for generating the DGIWG GML profiles

8.2.1 Analysis and extraction of geometry and topology information

For every DGIWG profile of ISO 19107 the included classes have been identified and compiled in STD-DS-09-228-ed1.0-Spatial_Profiles that is bundled within the same .zip folder as this document. These classes are mapped to their corresponding GML object elements.

8.2.2 Mapping of the extracted UML classes to their equivalent GML object elements

The mapping of UML Classes has been achieved by a simple Perl script using table D.2 of the ISO 19136. Table D.2 provides a mapping between conceptual UML classes implemented by ISO 19136 and the associated GML object elements, XML Schema types and GML property types.

While the DGIWG profiles are based on ISO 19107 concepts, not all of these concepts can be reflected directly in GML e.g. operations and boundary classes. This is reflected in the mapping tables below (Table 8.4 to Table 8.9). For additional information see ISO 19136 (D.3. and D.4). The 37 UML profiles described in Table 6.2 are mapped to 12 GML conformance levels (of type Lx_yD defined in Section 2 and in the following table).

	Spatial dimension	2D			3D			
	Primitive dimension	0d	1d	2d	0d	1d	2d	3d
Topological level	L1	L1_2D			L1_3D			
	L2	L2_2D			L2_3D			
	L3	L3_2D			L3_3D			
	L4	L4_2D			L4_3D			
	L5	L5_2D			L5_3D			
	L6	L6_2D			L6_3D			

Table 8.1 – Mappings of DGIWG UML profiles of ISO 19107 to DGIWG GML conformance levels

These 12 conformance levels are handled at the GML schema profile by 6 XSD documents (see the following table), each defining the GML geometric and topologic relevant classes. The 12 conformance levels are handled by only 6 schemas (see Table 8.2) because some restrictions expressed by the conformance levels could not be expressed through the GML schemas. Conformance levels L1_2D, L2_2D and L3_2D are handled by the same “2dGeometry” GML schema because these 3 levels allow the same geometric and topologic classes. The differences between these 3 levels are only visible at the GML data level (spaghetti data for L1_2D, no self intersection for L2_2D and primitives disjoint for L3_2D). These differences are discussed in section 6 - UML Profiles.

GML conformance levels	GML schema
L1_2D	2dGeometry
L2_2D	
L3_2D	
L1_3D	3dGeometry
L2_3D	
L3_3D	
L4_2D	2dComplex
L4_3D	3dComplex
L5_2D	2dTopology
L6_2D	
L5_3D	3dTopology
L6_3D	

Table 8.2 – Mappings of DGIWG GML conformance levels to DGIWG GML schemas

Application schemas using one or more of these GML profiles should specify them in the appInfo element (reference ISO 19136:2007 section 20.5). See Annex F – for further guidance.

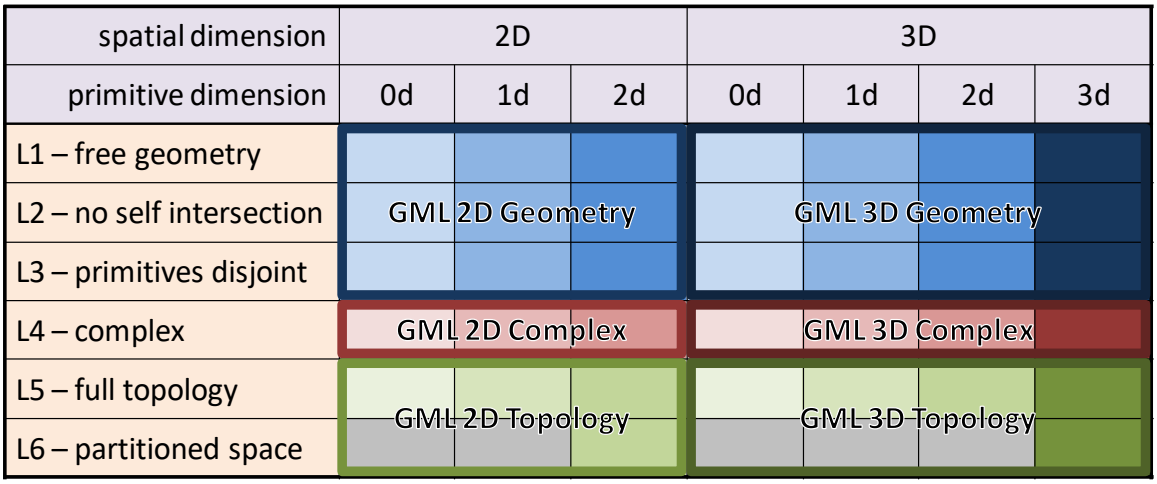


Table 8.3 – Mappings of UML profiles of ISO 19107 to DGIWG GML schemas

Table 8.4 through Table 8.9 show the mappings from the UML to the major GML profiles with the minor GML profiles separated by colour. The tables consist of the relevant UML classes extracted from the DGIWG Profiles of ISO 19107, the corresponding GML elements as input parameters for the XSLT subset tool of ISO 19136 and the mapping of both. The identified GML object elements for each profile are used as input parameters for the XSLT subset tool provided by the GML standard.

Surface in 2D → realization by gml:Polygon

Surface in 3D → realization by gml:Surface

NOTE Surfacepatch will NOT be used in 2D, but will be included in 3D

"To enable that "CompositeCurve" may be used in GML where in general a geometric primitive is expected, an abstract (and propertyless) object element "AbstractCurve" has been introduced and may be substituted by either "Curve", "OrientableCurve" or "CompositeCurve". The same mechanism is used with surfaces and solids. As a result, the "GM_OrientablePrimitive" class is not mapped to GML explicitly, however as this type is not instantiable, this does not impose any restrictions" [ISO 19136:2007 p.256].

The profiles do not include a mapping for the operations defined in ISO 19107 (e.g. the representative point operation). The reason for this is that operations are not mapped to GML in general. The GML standard defines that: "derived attributes are treated as operations and it is assumed that the derived attributes in the aggregate geometries will be derived from the data by the application handling the GML instances."

NOTE See ISO 19136:2007 p. 209 EXAMPLE 4: gml:PointPropertyType may be used directly as the content model for a property element ex:representativePoint.

Table 8.4 and Table 8.5 map L1, L2 and L3, 2D and 3D respectively to GML 2D Geometry and GML 3D Geometry. The difference between L1, L2, and L3 UML profiles is only in constraints on the data and not in the GML elements. L1 is free geometry where the interiors of primitives may intersect other primitives as well as self intersect. L2 excludes primitives self intersecting. L3 requires all primitives to be disjoint.

			L1.2D/L2.2D/L3.2D	Mapping	Parameter for XSLT Profiling
GML 2D Geometry 2d	GML 2D Geometry 1d	GML 2D Geometry 0d	GM_Object	GM_Object-->gml:AbstractGeometry	gml:AbstractGeometry
			GM_Primitive	GM_Primitive-->gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtive
			GM_Point	GM_Point-->gml:Point	gml:Point
			DirectPosition	DirectPosition-->gml:DirectPositionType	gml:DirectPositionType
			GM_Envelope		
			GM_Position	GM_Position-->gml:geometricPositionGroup	gml:geometricPositionGroup
			GM_Aggregate	GM_Aggregate-->gml:MultiGeometry	gml:MultiGeometry
			GM_MultiPrimitive	GM_MultiPrimitive-->gml:MultiGeometry	gml:MultiGeometry
			GM_MultiPoint	GM_MultiPoint-->gml:MultiPoint	gml:MultiPoint
			GM_OrientablePrimitive		
			GM_OrientableCurve	GM_OrientableCurve-->gml:OrientableCurve	gml:OrientableCurve
			GM_Curve	GM_Curve-->gml:Curve	gml:Curve
			GM_PointArray	GM_PointArray-->gml:geometricPositionListGroup	gml:geometricPositionListGroup
			GM_CurveInterpolation		
			GM_CurveSegment	GM_CurveSegment-->gml:AbstractCurveSegment	gml:AbstractCurveSegment
			GM_LineString	GM_LineString-->gml:LineStringSegment	gml:LineStringSegment
			GM_GeodesicString	GM_GeodesicString-->gml:GeodesicString	gml:GeodesicString
			GM_ArcString	GM_ArcString-->gml:ArcString	gml:ArcString
			GM_MultiCurve	GM_MultiCurve-->gml:MultiCurve	gml:MultiCurve
			GM_Boundary		
			GM_PrimitiveBoundary		
			GM_Ring	GM_Ring-->gml:Ring	gml:Ring
			GM_SurfaceBoundary		
			GM_OrientableSurface	GM_OrientableSurface-->gml:OrientableSurface	gml:OrientableSurface
			GM_Surface	GM_Surface-->gml:Surface	gml:Surface
			GM_MultiSurface	GM_MultiSurface-->gml:MultiSurface	gml:MultiSurface

Table 8.4 – Mapping for L1/L2/L3 2D to GML 2dGeometry schema

			L1.3D/L2.3D/L3.3D	Mapping	Parameter for XSLT Profiling
GML 3D Geometry 3d	GML 3D Geometry 2d	GML 3D Geometry 1d	GML 3D Geometry 0d	GM_Object	GM_Object-->gml:AbstractGeometry	gml:AbstractGeometry
				GM_Primitive	GM_Primitive-->gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtive
				GM_Point	GM_Point-->gml:Point	gml:Point
				DirectPosition	DirectPosition-->gml:DirectPositionType	gml:DirectPositionType
				GM_Envelope		
				GM_Position	GM_Position-->gml:geometricPositionGroup	gml:geometricPositionGroup
				GM_Aggregate	GM_Aggregate-->gml:MultiGeometry	gml:MultiGeometry
				GM_MultiPrimitive	GM_MultiPrimitive-->gml:MultiGeometry	gml:MultiGeometry
				GM_MultiPoint	GM_MultiPoint-->gml:MultiPoint	gml:MultiPoint
				GM_OrientablePrimitive		
				GM_OrientableCurve	GM_OrientableCurve-->gml:OrientableCurve	gml:OrientableCurve
				GM_Curve	GM_Curve-->gml:Curve	gml:Curve
				GM_PointArray	GM_PointArray-->gml:geometricPositionListGroup	gml:geometricPositionListGroup
				GM_CurveInterpolation		
				GM_CurveSegment	GM_CurveSegment-->gml:AbstractCurveSegment	gml:AbstractCurveSegment
				GM_LineString	GM_LineString-->gml:LineStringSegment	gml:LineStringSegment
				GM_GeodesicString	GM_GeodesicString-->gml:GeodesicString	gml:GeodesicString
				GM_ArcString	GM_ArcString-->gml:ArcString	gml:ArcString
				GM_MultiCurve	GM_MultiCurve-->gml:MultiCurve	gml:MultiCurve
				GM_Ring		
				GM_SurfaceBoundary		
				GM_OrientableSurface	GM_OrientableSurface-->gml:OrientableSurface	gml:OrientableSurface
				GM_Surface	GM_Surface-->gml:Surface	gml:Surface
				GM_PointGrid		
				GM_SurfaceInterpolation		
				GM_SurfacePatch	GM_SurfacePatch-->gml:AbstractSurfacePatch	gml:AbstractSurfacePatch
				GM_PolyhedralSurface	GM_PolyhedralSurface-->gml:PolyhedralSurface	gml:PolyhedralSurface
				GM_Polygon	GM_Polygon-->gml:PolygonPatch	gml:PolygonPatch
				GM_TriangulatedSurface	GM_TriangulatedSurface-->gml:TriangulatedSurface	gml:TriangulatedSurface
				GM_Triangle	GM_Triangle-->gml:Triangle	gml:Triangle
				GM_Tin	GM_Tin-->gml:Tin	gml:Tin
				GM_ParametricCurveSurface	GM_ParametricCurveSurface --> gml:AbstractParametricCurveSurface	gml:AbstractParametricCurveSurface
				GM_GriddedSurface	GM_GriddedSurface-->gml:AbstractGriddedSurface	gml:AbstractGriddedSurface
				GM_BilinearGrid		
				GM_MultiSurface	GM_MultiSurface-->gml:MultiSurface	gml:MultiSurface
				GM_Boundary		
				GM_PrimitiveBoundary		
				GM_Shell	GM_Shell-->gml:Shell	gml:Shell
				GM_SolidBoundary		
				GM_Solid	GM_Solid-->gml:Solid	gml:Solid
				GM_MultiSolid	GM_MultiSolid-->gml:MultiSolid	gml:MultiSolid

Table 8.5 – Mapping for L1/L2/L3 3D to GML 3dGeometry schema

Table 8.6 maps UML profile L4.2D to the GML 2dComplex schema.

			L4.2D	Mapping	Parameter for XSLT Profiling
GML 2D Complex 2d	GML 2D Complex 1d	GML 2D Complex 0d	GM_Object	GM_Object --> gml:AbstractGeometry	gml:AbstractGeometry
			GM_Primitive	GM_Primitive --> gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtive
			GM_Point	GM_Point --> gml:Point	gml:Point
			DirectPosition	DirectPosition --> gml:DirectPositionType	gml:DirectPositionType
			GM_Envelope		
			GM_Position	GM_Position --> gml:geometricPositionGroup	gml:geometricPositionGroup
			GM_Aggregate	GM_Aggregate --> gml:MultiGeometry	gml:MultiGeometry
			GM_MultiPrimitive	GM_MultiPrimitive --> gml:MultiGeometry	gml:MultiGeometry
			GM_MultiPoint	GM_MultiPoint --> gml:MultiPoint	gml:MultiPoint
			GM_Complex	GM_Complex --> gml:GeometricComplex	gml:GeometricComplex
			GM_Composite		
			GM_CompositePoint	GM_CompositePoint --> gml:Point	gml:Point
			GM_OrientablePrimitive		
			GM_OrientableCurve	GM_OrientableCurve --> gml:OrientableCurve	gml:OrientableCurve
			GM_Curve	GM_Curve --> gml:Curve	gml:Curve
			GM_PointArray	GM_PointArray --> gml:geometricPositionListGroup	gml:geometricPositionListGroup
			GM_CurveInterpolation		
			GM_CurveSegment	GM_CurveSegment --> gml:AbstractCurveSegment	gml:AbstractCurveSegment
			GM_LineString	GM_LineString --> gml:LineStringSegment	gml:LineStringSegment
			GM_GeodesicString	GM_GeodesicString --> gml:GeodesicString	gml:GeodesicString
			GM_ArcString	GM_ArcString --> gml:ArcString	gml:ArcString
			GM_MultiCurve	GM_MultiCurve --> gml:MultiCurve	gml:MultiCurve
			GM_CompositeCurve	GM_CompositeCurve --> gml:CompositeCurve	gml:CompositeCurve
			GM_OrientableSurface	GM_OrientableSurface --> gml:OrientableSurface	gml:OrientableSurface
			GM_Surface	GM_Surface --> gml:Surface	gml:Surface
			GM_Boundary		
			GM_PrimitiveBoundary		
			GM_Ring	GM_Ring --> gml:Ring	gml:Ring
			GM_SurfaceBoundary		
			GM_MultiSurface	GM_MultiSurface --> gml:MultiSurface	gml:MultiSurface
			GM_CompositeSurface	GM_CompositeSurface --> gml:CompositeSurface	gml:CompositeSurface

Table 8.6 – Mapping for L4 2D to GML 2D Complex

Table 8.7 maps UML profile L4.3D to the GML 3dComplex schema.

	L4.3D	Mapping	Parameter for XSLT Profiling
GML 3D Complex 0d	GM_Object	GM_Object --> gml:AbstractGeometry	gml:AbstractGeometry
	GM_Primitive	GM_Primitive --> gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtive
	GM_Point	GM_Point --> gml:Point	gml:Point
	DirectPosition	DirectPosition --> gml:DirectPositionType	gml:DirectPositionType
	GM_Envelope		
	GM_Position	GM_Position --> gml:geometricPositionGroup	gml:geometricPositionGroup
	GM_Aggregate	GM_Aggregate --> gml:MultiGeometry	gml:MultiGeometry
	GM_MultiPrimitive	GM_MultiPrimitive --> gml:MultiGeometry	gml:MultiGeometry
	GM_MultiPoint	GM_MultiPoint --> gml:MultiPoint	gml:MultiPoint
	GM_Complex	GM_Complex --> gml:GeometricComplex	gml:GeometricComplex
	GM_Composite		
	GM_CompositePoint	GM_CompositePoint --> gml:Point	gml:Point
GML 3D Complex 1d	GM_OrientablePrimitive		
	GM_OrientableCurve	GM_OrientableCurve --> gml:OrientableCurve	gml:OrientableCurve
	GM_Curve	GM_Curve --> gml:Curve	gml:Curve
	GM_PointArray	GM_PointArray --> gml:geometricPositionListGroup	gml:geometricPositionListGroup
	GM_CurveInterpolation		
	GM_CurveSegment	GM_CurveSegment --> gml:AbstractCurveSegment	gml:AbstractCurveSegment
	GM_LineString	GM_LineString --> gml:LineStringSegment	gml:LineStringSegment
	GM_GeodesicString	GM_GeodesicString --> gml:GeodesicString	gml:GeodesicString
	GM_ArcString	GM_ArcString --> gml:ArcString	gml:ArcString
	GM_MultiCurve	GM_MultiCurve --> gml:MultiCurve	gml:MultiCurve
	GM_CompositeCurve	GM_CompositeCurve --> gml:CompositeCurve	gml:CompositeCurve
GML 3D Complex 2d	GM_Ring		
	GM_SurfaceBoundary		
	GM_OrientableSurface	GM_OrientableSurface --> gml:OrientableSurface	gml:OrientableSurface
	GM_Surface	GM_Surface --> gml:Surface	gml:Surface
	GM_PointGrid		
	GM_SurfaceInterpolation		
	GM_SurfacePatch	GM_SurfacePatch --> gml:AbstractSurfacePatch	gml:AbstractSurfacePatch
	GM_PolyhedralSurface	GM_PolyhedralSurface --> gml:PolyhedralSurface	gml:PolyhedralSurface
	GM_Polygon	GM_Polygon --> gml:PolygonPatch	gml:PolygonPatch
	GM_TriangulatedSurface	GM_TriangulatedSurface --> gml:TriangulatedSurface	gml:TriangulatedSurface
	GM_Triangle	GM_Triangle --> gml:Triangle	gml:Triangle
	GM_Tin	GM_Tin --> gml:Tin	gml:Tin
	GM_ParametricCurveSurface	GM_ParametricCurveSurface --> gml:AbstractParametricCurveSurface	gml:AbstractParametricCurveSurface
	GM_GriddedSurface	GM_GriddedSurface --> gml:AbstractGriddedSurface	gml:AbstractGriddedSurface
	GM_BilinearGrid		
	GM_MultiSurface	GM_MultiSurface --> gml:MultiSurface	gml:MultiSurface
	GM_CompositeSurface	GM_CompositeSurface --> gml:CompositeSurface	gml:CompositeSurface
GML 3D Complex 3d	GM_Boundary		
	GM_PrimitiveBoundary		
	GM_Shell	GM_Shell --> gml:Shell	gml:Shell
	GM_SolidBoundary		
	GM_Solid	GM_Solid --> gml:Solid	gml:Solid
	GM_MultiSolid	GM_MultiSolid --> gml:MultiSolid	gml:MultiSolid
	GM_CompositeSolid	GM_CompositeSolid --> gml:CompositeSolid	gml:CompositeSolid

Table 8.7 – Mapping for L4 3D to GML 3D Complex

Table 8.8 and Table 8.9 map L5 and L6, 2D and 3D respectively to GML 2dTopology and GML 3dTopology. The difference between L5 and L6 UML profiles is only in constraints on the data and not in the GML elements. L5 allows holes in the geometric data coverage. L6 requires a full tessellation of the surface in 2D and the space in 3D.

			L5.2D/L6.2D	Mapping	Parameter for XSLT Profiling
GML 2D Topology 2d	GML 2D Topology 1d	GML 2D Topology 0d	GM_Object	GM_Object --> gml:AbstractGeometry	gml:AbstractGeometry
			GM_Primitive		
			GM_Point	GM_Point --> gml:Point	gml:Point
			DirectPosition	DirectPosition --> gml:DirectPositionType	gml:DirectPositionType
			GM_Envelope		
			GM_Position	GM_Position --> gml:geometricPositionGroup	gml:geometricPositionGroup
			GM_Aggregate	GM_Aggregate --> gml:MultiGeometry	gml:MultiGeometry
			GM_MultiPrimitive	GM_MultiPrimitive --> gml:MultiGeometry	gml:MultiGeometry
			GM_MultiPoint	GM_MultiPoint --> gml:MultiPoint	gml:MultiPoint
			GM_Complex	GM_Complex --> gml:GeometricComplex	gml:GeometricComplex
			GM_Composite		
			GM_CompositePoint	GM_CompositePoint --> gml:Point	gml:Point
			TP_Object	TP_Object --> gml:AbstractTopology	gml:AbstractTopology
			TP_Boundary		
			TP_PrimitiveBoundary		
			TP_Primitive	GM_Primitive --> gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtive
			TP_DirectedTopo		
			TP_Node	TP_Node --> gml:Node	gml:Node
			TP_DirectedNode	TP_DirectedNode --> gml:DirectedNodePropertyType	gml:DirectedNodePropertyType
			TP_Complex	TP_Complex --> gml:TopoComplex	gml:TopoComplex
			GM_CurveBoundary		
			GM_OrientablePrimitive		
			GM_OrientableCurve	GM_OrientableCurve --> gml:OrientableCurve	gml:OrientableCurve
			GM_Curve	GM_Curve --> gml:Curve	gml:Curve
			GM_PointArray	GM_PointArray --> gml:geometricPositionListGroup	gml:geometricPositionListGroup
			GM_CurveInterpolation		
			GM_CurveSegment	GM_CurveSegment --> gml:AbstractCurveSegment	gml:AbstractCurveSegment
			GM_LineString	GM_LineString --> gml:LineStringSegment	gml:LineStringSegment
			GM_GeodesicString	GM_GeodesicString --> gml:GeodesicString	gml:GeodesicString
			GM_ArcString	GM_ArcString --> gml:ArcString	gml:ArcString
			GM_MultiCurve	GM_MultiCurve --> gml:MultiCurve	gml:MultiCurve
			GM_CompositeCurve	GM_CompositeCurve --> gml:CompositeCurve	gml:CompositeCurve
			TP_EdgeBoundary		
			TP_Edge	TP_Edge --> gml:Edge	gml:Edge
			TP_DirectedEdge	TP_DirectedEdge --> gml:DirectedEdgePropertyType	gml:DirectedEdgePropertyType
			TP_Expression		
			GM_OrientableSurface	GM_OrientableSurface --> gml:OrientableSurface	gml:OrientableSurface
			GM_Surface	GM_Surface --> gml:Surface	gml:Surface
			GM_Boundary		
			GM_PrimitiveBoundary		
			GM_Ring	GM_Ring --> gml:Ring	gml:Ring
			GM_SurfaceBoundary		
			GM_MultiSurface	GM_MultiSurface --> gml:MultiSurface	gml:MultiSurface
			GM_CompositeSurface	GM_CompositeSurface --> gml:CompositeSurface	gml:CompositeSurface
			TP_Face	TP_Face --> gml:Face	gml:Face
			TP_DirectedFace	TP_DirectedFace --> gml:DirectedFacePropertyType	gml:DirectedFacePropertyType
			TP_FaceBoundary		
			TP_Ring		

Table 8.8 – Mapping for L5/L6 2D to GML 2dTopology schema

	L5.3D/L6.3D	Mapping	Parameter for XSLT Profiling
GML 3D Topology 0d	GM_Object	GM_Object --> gml:AbstractGeometry	gml:AbstractGeometry
	GM_Primitive	GM_Primitive --> gml:AbstractGeometricPrimtive	gml:AbstractGeometricPrimtive
	GM_Point	GM_Point --> gml:Point	gml:Point
	DirectPosition	DirectPosition --> gml:DirectPositionType	gml:DirectPositionType
	GM_Envelope		
	GM_Position	GM_Position --> gml:geometricPositionGroup	gml:geometricPositionGroup
	GM_Aggregate	GM_Aggregate --> gml:MultiGeometry	gml:MultiGeometry
	GM_MultiPrimitive	GM_MultiPrimitive --> gml:MultiGeometry	gml:MultiGeometry
	GM_MultiPoint	GM_MultiPoint --> gml:MultiPoint	gml:MultiPoint
	GM_Complex	GM_Complex --> gml:GeometricComplex	gml:GeometricComplex
	GM_Composite		
	GM_CompositePoint	GM_CompositePoint --> gml:Point	gml:Point
	TP_Object	TP_Object --> gml:AbstractTopology	gml:AbstractTopology
	TP_Boundary		
	TP_PrimitiveBoundary		
	TP_Primitive		
	TP_DirectedTopo		
	TP_Node	TP_Node --> gml:Node	gml:Node
	TP_DirectedNode	TP_DirectedNode --> gml:DirectedNodePropertyType	gml:DirectedNodePropertyType
	TP_Complex	TP_Complex --> gml:TopoComplex	gml:TopoComplex
GML 3D Topology 1d	GM_CurveBoundary		
	GM_OrientablePrimitive		
	GM_OrientableCurve	GM_OrientableCurve --> gml:OrientableCurve	gml:OrientableCurve
	GM_Curve	GM_Curve --> gml:Curve	gml:Curve
	GM_PointArray	GM_PointArray --> gml:geometricPositionListGroup	gml:geometricPositionListGroup
	GM_CurveInterpolation		
	GM_CurveSegment	GM_CurveSegment --> gml:AbstractCurveSegment	gml:AbstractCurveSegment
	GM_LineString	GM_LineString --> gml:LineStringSegment	gml:LineStringSegment
	GM_GeodesicString	GM_GeodesicString --> gml:GeodesicString	gml:GeodesicString
	GM_ArcString	GM_ArcString --> gml:ArcString	gml:ArcString
	GM_MultiCurve	GM_MultiCurve --> gml:MultiCurve	gml:MultiCurve
	GM_CompositeCurve	GM_CompositeCurve --> gml:CompositeCurve	gml:CompositeCurve
	TP_EdgeBoundary		
	TP_Edge	TP_Edge --> gml:Edge	gml:Edge
	TP_DirectedEdge	TP_DirectedEdge --> gml:DirectedEdgePropertyType	gml:DirectedEdgePropertyType
	TP_Expression		
GML 3D Topology 2d	GM_Boundary		
	GM_PrimitiveBoundary		
	GM_Ring	GM_Ring --> gml:Ring	gml:Ring
	GM_SurfaceBoundary		
	GM_OrientableSurface	GM_OrientableSurface --> gml:OrientableSurface	gml:OrientableSurface
	GM_Surface	GM_Surface --> gml:Surface	gml:Surface
	GM_PointGrid		
	GM_SurfaceInterpolation		
	GM_SurfacePatch	GM_SurfacePatch --> gml:AbstractSurfacePatch	gml:AbstractSurfacePatch
	GM_PolyhedralSurface	GM_PolyhedralSurface --> gml:PolyhedralSurface	gml:PolyhedralSurface
	GM_Polygon	GM_Polygon --> gml:PolygonPatch	gml:PolygonPatch
	GM_TriangulatedSurface	GM_TriangulatedSurface --> gml:TriangulatedSurface	gml:TriangulatedSurface
	GM_Triangle	GM_Triangle --> gml:Triangle	gml:Triangle
	GM_Tin	GM_Tin --> gml:Tin	gml:Tin
	GM_ParametricCurveSurface	GM_ParametricCurveSurface --> gml:AbstractParametricCurveSurface	gml:AbstractParametricCurveSurface
	GM_GriddedSurface	GM_GriddedSurface --> gml:AbstractGriddedSurface	gml:AbstractGriddedSurface
	GM_BilinearGrid		
	GM_MultiSurface	GM_MultiSurface --> gml:MultiSurface	gml:MultiSurface
	GM_CompositeSurface	GM_CompositeSurface --> gml:CompositeSurface	gml:CompositeSurface
	TP_FaceBoundary		
	TP_Ring		
	TP_Face	TP_Face --> gml:Face	gml:Face
	TP_DirectedFace	TP_DirectedFace --> gml:DirectedFacePropertyType	gml:DirectedFacePropertyType
GML 3D Topology 3d	GM_Shell	GM_Shell --> gml:Shell	gml:Shell
	GM_SolidBoundary		
	GM_Solid	GM_Solid --> gml:Solid	gml:Solid
	GM_MultiSolid	GM_MultiSolid --> gml:MultiSolid	gml:MultiSolid
	GM_CompositeSolid	GM_CompositeSolid --> gml:CompositeSolid	gml:CompositeSolid
	TP_SolidBoundary		
	TP_Shell		
	TP_Solid	TP_Solid --> gml:TopoSolid	gml:TopoSolid
	TP_DirectedSolid	TP_DirectedSolid --> gml:DirectedTopoSolidPropertyType	gml:DirectedTopoSolidPropertyType

Table 8.9 – Mapping for L5/L6 3D to GML 3dTopology schema

8.3 Generating the different profiles with the GML subset tool provided by the GML standard

The Informative Annex G of ISO 19136 contains an XSLT reference implementation of a GML schema subset tool. The tool consists of three XSLT style sheets that create an automated subsetting of the GML schema:

- depends.xslt
- utility.xslt
- gmlSubset.xslt

The depends.xslt and the utility.xslt style sheet transform gml.xsd into a file called gml.dep, which includes all GML object elements and their dependencies. The gml.dep file is used as the basis for the generation of different profiles of the GML schema.

To generate a GML profile, the “wanted” parameters of the gmlSubset.xslt style sheet have to be adapted. The “wanted” parameter contains a comma separated list of GML object elements for the profile. The list has to end with a comma. These have been identified by mapping the UML classes of the DGIWG Spatial Schema Profiles of ISO 19107 to their equivalent GML object elements.

Example for L1 2d2D:

```
<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,
</xsl:param>
```

Example for L13D3d:

```
<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:Solid,gml:SolidPropertyType,gml:Shell,gml:MultiSolid,gml:MultiSolidPropertyType,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,
</xsl:param>
```

Example for L32D2d:

```
<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:OrientableCurve,gml:OrientableSurface,
</xsl:param>
```

Example for L33D3d:

```
<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:Solid,gml:SolidPropertyType,gml:Shell,gml:MultiSolid,gml:MultiSolidPropertyType,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,
</xsl:param>
```

Example for L42D2d:

<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,gml:CompositeCurve,gml:OrientableCurve,gml:GeometricComplex,gml:GeometricComplexPropertyType,gml:CompositeSurface,gml:OrientableSurface,
</xsl:param>

Example for L43D3d:

<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:Solid,gml:SolidPropertyType,gml:Shell,gml:MultiSolid,gml:MultiSolidPropertyType,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,gml:CompositeCurve,gml:OrientableCurve,gml:GeometricComplex,gml:GeometricComplexPropertyType,gml:CompositeSurface,gml:OrientableSurface,gml:CompositeSolid,
</xsl:param>

Example for L5_2D_2d:

<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,gml:CompositeCurve,gml:OrientableCurve,gml:GeometricComplex,gml:GeometricComplexPropertyType,gml:CompositeSurface,gml:OrientableSurface,gml:TopoComplex,gml:TopoComplexPropertyType,gml:Node,gml:directedNode,gml:Edge,gml:directedEdge,gml:Face,gml:directedFace,
</xsl:param>

Example for L53D3d:

<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:Solid,gml:SolidPropertyType,gml:Shell,gml:MultiSolid,gml:MultiSolidPropertyType,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,gml:CompositeCurve,gml:OrientableCurve,gml:GeometricComplex,gml:GeometricComplexPropertyType,gml:CompositeSurface,gml:OrientableSurface,gml:CompositeSolid,gml:TopoComplex,gml:TopoComplexPropertyType,gml:Node,gml:directedNode,gml:Edge,gml:directedEdge,gml:Face,gml:directedFace,gml:TopoSolid, gml:directedTopoSolid,
</xsl:param>

Example for L62D2d:

<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:

AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,gml:CompositeCurve,gml:OrientableCurve,gml:GeometricComplex,gml:GeometricComplexPropertyType,gml:CompositeSurface,gml:OrientableSurface,gml:TopoComplex,gml:TopoComplexPropertyType,gml:Node,gml:directedNode,gml:Edge,gml:directedEdge,gml:Face,gml:directedFace,
</xsl:param>

Example for L63D3d:

<xsl:param name="wanted">
gml:Point,gml:PointPropertyType,gml:MultiPoint,gml:MultiPointPropertyType,gml:Curve,gml:LineStringSegment,gml:LineString,gml:CurvePropertyType,gml:MultiCurve,gml:MultiCurvePropertyType,gml:Surface,gml:PolygonPatch,gml:Polygon,gml:SurfacePropertyType,gml:LinearRing,gml:Ring,gml:MultiSurface,gml:MultiSurfacePropertyType,gml:ArcString,gml:GeodesicString,gml:MultiGeometry,gml:geometricPositionListGroup,gml:Solid,gml:SolidPropertyType,gml:Shell,gml:MultiSolid,gml:MultiSolidPropertyType,gml:AbstractGriddedSurface,gml:AbstractParametricCurveSurface,gml:PolyhedralSurface,AbstractGeometricPrimtive,gml:Tin,gml:Triangle,gml:TriangulatedSurface,gml:OrientableCurve,gml:OrientableSurface,gml:CompositeCurve,gml:OrientableCurve,gml:GeometricComplex,gml:GeometricComplexPropertyType,gml:CompositeSurface,gml:OrientableSurface,gml:CompositeSolid,gml:TopoComplex,gml:TopoComplexPropertyType,gml:Node,gml:directedNode,gml:Edge,gml:directedEdge,gml:Face,gml:directedFace,gml:TopoSolid, gml:directedTopoSolid,
</xsl:param>

By transforming gml.dep with gmlSubset.xslt, using a stylesheet processor the relevant profile is generated automatically.

8.4 Adapting the generated profiles according to the specializations defined by the DGIWG Spatial Schema Profiles of ISO 19107

The GML subsetting tool does not catch all requirements specified automatically. The adaption of the GML profiles to these requirements has to be done manually. This is done by applying the requirements defined in the Abstract Test Suite.

9 GML application schemas build on DGIWG GML profiles

9.1 Introduction

This section contains discussion and schema coding patterns that apply to all DGIWG GML compliance levels defined in this document. It provides requirements on how to use the DGIWG GML profiles within a GML application schema.

9.2 Importing the compliance level schema

This document defines twelve compliance levels as described in section 2 - Conformance2. In order for a client to be able to properly interpret a schema, it shall be able to identify the compliance level of the schema. For this purpose, the compliance level schema shall first be imported. The following XML fragment imports the schema that declares the ComplianceLevel element used to declare the compliance level of an application schema :

```
<xsd:import
    namespace="http://www.dgiwg.org/gml/3.2/profiles/spatial/1.0/"
    schemaLocation=
"http://schemas.dgiwg.org/gml/3.2/profiles/spatial/1.0/gmldgiwgspLevels.xsd"
/>
```

9.3 Identifying the DGIWG GML compliance level

Once the compliance level schema, an XML annotation shall be used for indentifying the compliance level. The annotation shall be placed at the top most nesting level of the schema such that the XPath expression:

/xsd:schema/xsd:annotation/xsd:appinfo/gmlsp:ComplianceLevel

The following schema fragment shows how this annotation shall be declared in an application schema:

```
<xsd:annotation>
    <xsd:appinfo
    source="http://schemas.dgiwg.org/gml/3.2/spatial/1.0/gmldgiwgspLevels.xsd"
    >
        <gmlsp:ComplianceLevel>Lx_yD</gmlsp:ComplianceLevel>
        <gmlsp:GMLProfileSchema>http://schemas.dgiwg.org/gml/3.2/spatial/1.0/
        gmlxsdSchema.xsd</gmlsp:GMLProfileSchema>
    </xsd:appinfo>
</xsd:annotation>
```

Lx_yD refers to one of the conformance level defines in section 2 - Conformance. So it shall be replaced by one of the twelve following values : L1_2D, L2_2D, L3_2D, L4_2D, L5_2D, L6_2D, L1_3D, L2_3D, L3_3D, L4_3D, L5_3D, L6_3D.

gmlxsdSchema.xsd refers to the corresponding GML schema, according to the mapping Table 8.2 – Mappings of DGIWG GML conformance levels to DGIWG GML schemas. It shall be then replaced by one of the six following values 2dGeometry.xsd, 3dGeometry.xsd, 2dComplex.xsd, 3dComplex.xsd, 2dTopology.xsd, 3dTopology.xsd.

10 GML data realizations of DGIWG GML profiles

10.1 Introduction

This section contains discussion on GML data compliance against the twelve levels of conformance defined in section 2 - Conformance. It provides requirements on GML data. In particular it underlines the differences between data that relies on the same GML schema but which comply to different conformance levels.

First GML data which claim conformance to any DGIWG GML conformance level (Lx_yD) shall be validated by its GML application schema. So it directly implies that these GML data only use the GML geometric/topologic allowed by the appropriate DGIWG GML conformance level.

10.2 GML data using geometric profiles

This sub-section deals with GML data compliant with L1_2D, L2_2D, L3_2D (and L1_3D, L2_2D, L3_3D).

Compliance levels are build in a hierarchical way, such as a constraint required for a certain level will be required for a higher level too. That means that each requirement applying to Lx_yD also applies to Lx'_y'D as soon as x'>x.

10.2.1 GML data using free geometry (L1_2D, L1_3D)

GML data which claim conformance to L1_2D and L1_3D do not have any further requirement. They are commonly called spaghetti data.

10.2.2 GML data using no self intersection geometry (L2_2D, L2_3D)

In addition, for GML data which claim conformance to L2_2D or L2_3D, primitives shall be simple, i.e. do not self intersect.

Note : other requirement to satisfy is 10.1.

10.2.3 GML data using primitive disjoint geometry (L3_2D, L3_3D)

In addition, for GML data which claim conformance to L3_2D or L3_3D, primitives shall be disjoint, i.e. do not intersect each other. The explanation of what is exactly disjoint is available in Annex G – Disjoint.

Note : other requirements to satisfy are 10.1 and 10.2.2.

10.2.4 GML data using complex geometry (L4_2D, L4_3D)

In addition, for GML data which claim conformance to L4_2D or L4_3D, all boundary primitives shall exist (collection of primitives forms a complex).

Note : other requirements to satisfy are 10.1, 10.2.2 and 10.2.3.

10.2.5 GML data using full topology geometry (L5_2D, L5_3D)

In addition, for GML data which claim conformance to L5_2D or L5_3D, all boundary and coboundary relationships shall exist.

Note : other requirements to satisfy are 10.1, 10.2.2, 10.2.3 and 10.2.4.

10.2.6 GML data using partitioned space (L6_2D, L6_3D)

In addition, for GML data which claim conformance to L6_2D or L6_3D, space shall be partitioned.

Note : other requirements to satisfy are 10.1, 10.2.2, 10.2.3, 10.2.4 and 10.2.5.

Annex A – Abstract test suite

A.1 Introduction

This Annex provides abstract test suites for the UML profiles of ISO 19107 and the GML realization of these profiles. The abstract tests are hierarchical such that each test includes all tests of the preceding level. The first part of the tests consists of *General Tests* (A.2); each general test concerns one specialization (from S1 to S34). These tests are then used for the appropriate DGIWG profiles in the following parts (from A.3 for L2.2D.0d profile to A.39 for L6.3D.3d profile). Each of these tests are subdivided into two parts: first part for UML requirement and second part for GML requirements.

Table A.1 references the applicable paragraphs and requirements for testing the different profile levels according to this document

Level	2D.0d	2D.1d	2D.2d	3D.0d	3D.1d	3D.2d	3D.3d
L1	A.3	A.4	A.5	A.6	A.7	A.8	A.9
L2	A.10	A.11	A.12	A.13	A.14	A.15	A.16
L3	A.17	A.18	A.19	A.20	A.21	A.22	A.23
L4	A.24	A.25	A.26	A.27	A.28	A.29	A.30
L5	A.31	A.32	A.33	A.34	A.35	A.36	A.37
L6	--	--	A.38	--	--	--	A.39

Table A.1 - Requirements sections for each DGIWG Profile of ISO 19107 (UML and GML requirements)

A.2 General Tests

A.2.1 GM_Object RepresentativePoint (S1)

a) Test Purpose: Verify that the instantiation of GM_Object in an application schema includes the optional attribute *representativePoint*.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.2 GM_Object Envelope operation (S2)

a) Test Purpose: Verify that the instantiation of GM_Object in an application schema includes the optional attribute *envelope*.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.3 GM_SurfaceBoundary, GM_Ring Support GM_Polygon (S3)

a) Test Purpose: Verify that in an application schema the GM_SurfaceBoundary and GM_Ring are only used to support GM_Polygon.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
Test Type: Capability Test

A.2.4 GM_Primitives simple (S4)

a) Test Purpose: Verify that that all GM_Primitives are simple.
b) Test Method: Inspect the data.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.5 GM_Primitives disjoint (S5)

a) Test Purpose: Verify that the all GM_Primitives are disjoint.
b) Test Method: Inspect the data.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.6 GM_Primitives InteriorTo association constraint (S6)

a) Test Purpose: Verify that in an application schema the multiplicities of the InteriorTo association roles, containedPrimitive and containingPrimitive, are constrained to [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.7 GM_Primitives InteriorTo association used like the TP_Primitive::IsolatedIn association (S7)

a) Test Purpose: Verify that in an application schema the InteriorTo association only associates GM_Primitives with a dimensional difference greater than one and that the multiplicity of the superElement association role is constrained to [0..1].
b) Test Method: Inspect the data.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.8 GM_Primitive Oriented association (S8)

a) Test Purpose: Verify that in an application schema the multiplicity of the proxy role of the Oriented association between GM_Primitive and GM_OrientablePrimitive is constrained [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.9 GM_OrientableCurve is abstract and not instantiable (S9)

a) Test Purpose: Verify that in an application schema the implementation of GM_OrientableCurve is not instantiable.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.10 GM_OrientableSurface is abstract and not instantiable (S10)

a) Test Purpose: Verify that in an application schema the implementation of GM_OrientableSurface is not instantiable.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.11 GM_OrientableCurve boundary operation is changed in GM_Curve (S11)

a) Test Purpose: Verify that the instantiation of GM_Curve in an application schema includes a composition relationship to GM_CurveBoundary.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.12 GM_OrientableSurface boundary operation is changed in GM_Surface (S12)

a) Test Purpose: Verify that the instantiation of GM_Surface in an application schema includes a composition relationship to GM_SurfaceBoundary.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.13 GM_CurveSegment and GM_Curve segmentation constraint (S13)

a) Test Purpose: Verify that in an application schema the multiplicity of the curve role of the Segmentation association between GM_CurveSegment and GM_Curve is constrained to [1].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.14 GM_Surface boundary and GM_SurfacePatch is not used (S14)

a) Test Purpose: Verify that in an application schema GM_Surface is described by its boundary composition association to GM_SurfaceBoundary and that the multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained to [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.15 GM_Surface patch interior is described by GM_SurfacePatches, Segmentation association constraint (S15)

a) Test Purpose: Verify that in an application schema the multiplicity of the patch role of the Segmentation association between GM_Surface and GM_SurfacePatch is constrained to [1..n].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.16 GM_Solid boundary operation composition (S16)

a) Test Purpose: Verify that the instantiation of GM_Solid in an application schema includes a composition relationship to GM_SolidBoundary.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.17 Coordinate Reference System association (S17)

a) Test Purpose: Verify that in an application schema the multiplicity of the CRS role of the Coordinate Reference System association between DirectPosition and SC_CRS is constrained to [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.18 GM_Position union constraint (S18)

a) Test Purpose: Verify that in an application schema the union GM_Position always uses the direct variant and that the indirect variant is not used.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.19 GM_CurveInterpolation constraint (S19)

a) Test Purpose: Verify that in an application schema the GM_CurveInterpolation code list is constrained to the values “linear”, “geodesic”, and "circularArc3Points".
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.20 GM_CurveSegment constraint (S20)

a) Test Purpose: Verify that in an application schema the multiplicities of all three numDerivatives attributes of GM_CurveSegment are constrained to [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.21 GM_SurfaceInterpolation constraint (S21)

a) Test Purpose: Verify that in an application schema the GM_SurfaceInterpolation code list is constrained to the values “none”, “planar”, and “parametricCurve”.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.22 GM_SurfacePatch and GM_Surface constraint (S22)

a) Test Purpose: Verify that in an application schema the multiplicity of the surface role of the Segmentation association between GM_SurfacePatch and GM_Surface is constrained to [1].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.23 GM_SurfacePatch constraint (S23)

a) Test Purpose: Verify that in an application schema the multiplicity of the numDerivativesOnBoundary attribute of GM_SurfacePatch is constrained to [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.24 GM_Polygon constraint (S24)

a) Test Purpose: Verify that in an application schema the multiplicity of the spanningSurface attribute of GM_Polygon is constrained to [0].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.25 GM_Complex constraint (S25)

a) Test Purpose: Verify that the instantiation of GM_Complex in an application schema includes the optional Boolean attribute isMaximal.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.26 TP_Primitive and GM_Primitive association constraint (S26)

a) Test Purpose: Verify that in an application schema the multiplicity of the geometry role of the Realization association between TP_Primitive and GM_Primitive is constrained to [1] and the multiplicity of the topology role of the Realization association between GM_Primitive and TP_Primitive is constrained to [1].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.27 TP_Primitive association constraint (S27)

a) Test Purpose: Verify that the instantiation of TP_Primitive in an application schema includes the association role *maximalComplex*.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.28 TP_Node and TP_DirectedEdge CoBoundary association specialization (S28)

a) Test Purpose: Verify that in an application schema the spoke role of the CoBoundary association between TP_Node and TP_DirectedEdge is implemented as a Circular Sequence.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.29 TP_Edge and TP_DirectedFace CoBoundary multiplicity association constraint to [0..2] (S29)

a) Test Purpose: Verify that in an application schema the multiplicity of the spoke role of the CoBoundary association between TP_Edge and TP_DirectedFace is constrained to [0..2].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.30 TP_Edge and TP_DirectedFace CoBoundary multiplicity association constraint to [2] (S30)

a) Test Purpose: Verify that in an application schema the multiplicity of the spoke role of the CoBoundary association between TP_Edge and TP_DirectedFace is constrained to [2].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.31 TP_Face and TP_DirectedSolid CoBoundary multiplicity association constraint to [2] (S31)

a) Test Purpose: Verify that in an application schema the multiplicity of the spoke role of the CoBoundary association between TP_Face and TP_DirectedSolid is constrained to [2].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.32 TP_Complex association (S32)

a) Test Purpose: Verify that the instantiation of TP_Complex in an application schema includes the association role *maximalComplex*.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.33 TP_Complex constraint (S33)

a) Test Purpose: Verify that the instantiation of TP_Complex in an application schema includes the optional Boolean attribute isMaximal.
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.2.34 TP_Complex and GM_Complex association constraint (S34)

a) Test Purpose: Verify that in an application schema the multiplicity of the geometry role of the Realization association between TP_Complex and GM_Complex is constrained to [1] and that the multiplicity of the topology role of the Realization association between GM_Complex and TP_Complex is constrained to [1].
b) Test Method: Inspect the implementation.
c) Reference: DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.3 Abstract test suite for Profile L1.2D.0d

A.3.1 Abstract test suite for UML of L1.2D.0d

A.3.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.3.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.3.1.3 Abstract test suite for the DGIWG specializationsand constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.6, A.2.17, A.2.18
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.3.2 Abstract test suite for GML of L1.2D.0d

A.3.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.3.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.3.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.3.1
d) Test Type: Capability Test

A.4 Abstract test suite for Profile L1.2D.1d

A.4.1 Abstract test suite for UML of L1.2D.1d

A.4.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.1, 6.3.5, 6.3.13, 6.3.14.1, 6.3.16, 6.4.1, 6.4.6, 6.4.8 - 6.4.31, and 6.5.5.
d) Test Type: Capability Test

A.4.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.3.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.3.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.4.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.6, A.2.8, A.2.9, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.4.2 Abstract test suite for GML of L1.2D.1d

A.4.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 11.3.2, 11.3.3, 20, 21.12,
d) Test Type: Capability Test

A.4.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.4.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.4.1
d) Test Type: Capability Test

A.5 Abstract test suite for Profile L1.2D.2d

A.5.1 Abstract test suite for UML of L1.2D.2d

A.5.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.3.6, 6.3.7, 6.3.10.4, 6.3.15, 6.3.17.1, 6.3.17.3, 6.4.6, 6.4.32-6.4.48, and 6.5.6
d) Test Type: Capability Test

A.5.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.3.2.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_MultiSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.3.2.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.5.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.6, A.2.8, A.2.9, A.2.10, A.2.12, A.2.13, A.2.14, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.5.2 Abstract test suite for GML of L1.2D.2d

A.5.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 10.5.1, 10.5.2, 10.5.4 - 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 - 10.5.11.6, 11.3.2, 11.3.3, 11.3.4, 20, 21.12
d) Test Type: Capability Test

A.5.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.5.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.5.1
d) Test Type: Capability Test

A.6 Abstract test suite for Profile L1.3D.0d

A.6.1 Abstract test suite for UML of L1.3D.0d

A.6.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.6.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.6.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.6, A.2.17, A.2.18
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.6.2 Abstract test suite for GML of L1.3D.0d

A.6.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.6.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.6.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.6.1
d) Test Type: Capability Test

A.7 Abstract test suite for Profile L1.3D.1d

A.7.1 Abstract test suite for UML of L1.3D.1d

A.7.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.1, 6.3.5, 6.3.13, 6.3.14.1, 6.3.16, 6.4.1, 6.4.6, 6.4.8 - 6.4.31, and 6.5.5.
d) Test Type: Capability Test

A.7.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.6.1.2 and in addition implements ISO 19107 classes GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.6.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.7.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.6, A.2.8, A.2.9, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.7.2 Abstract test suite for GML of L1.3D.1d

A.7.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 11.3.2, 11.3.3, 20, 21.12,
d) Test Type: Capability Test

A.7.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.7.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.7.1
d) Test Type: Capability Test

A.8 Abstract test suite for Profile L1.3D.2d

A.8.1 Abstract test suite for UML of L1.3D.2d

A.8.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.3.6, 6.3.7, 6.3.10.4, 6.3.15, 6.3.17.1, 6.3.17.3, 6.4.6, 6.4.32 - 6.4.48, and 6.5.6
d) Test Type: Capability Test

A.8.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.7.1.2 in this document and in addition implements ISO 19107 classes: GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_PointGrid, GM_SurfaceInterpolation, GM_SurfacePatch, GM_PolyhedralSurface, GM_Polygon, GM_TriangulatedSurface, GM_Triangle, GM_Tin, GM_ParametricCurveSurface, GM_GriddedSurface, GM_BilinearGrid, GM_MultiSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.7.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.8.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.6, A.2.8, A.2.9, A.2.10, A.2.13, A.2.15, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.8.2 Abstract test suite for GML of L1.3D.2d

A.8.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.

c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 10.5.1, 10.5.2, 10.5.4 - 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 - 10.5.11.6, 11.3.2, 11.3.3, 11.3.4, 20, 21.12
d) Test Type: Capability Test

A.8.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.8.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.8.1
d) Test Type: Capability Test

A.9 Abstract test suite for Profile L1.3D.3d

A.9.1 Abstract test suite for UML of L1.3D.3d

A.9.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.4 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, 6.3.8, 6.3.9, 6.3.10.4, 6.3.18.1 and 6.5.7
d) Test Type: Capability Test

A.9.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.8.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Shell, GM_SolidBoundary, GM_Solid, GM_MultiSolid,
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.8.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.9.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.6, A.2.8, A.2.9, A.2.10, A.2.13, A.2.15, A.2.16, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.9.2 Abstract test suite for GML of L1.3D.3d

A.9.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.4 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, 10.6.1, 10.6.2, 10.6.4 -10.6.6, 11.3.5, 20, 21.12
d) Test Type: Capability Test

A.9.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.9.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.9.1
d) Test Type: Capability Test

A.10 Abstract test suite for Profile L2.2D.0d

A.10.1 Abstract test suite for UML of L2.2D.0d

A.10.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.

c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.10.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position , GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.10.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.4, A.2.6, A.2.17, A.2.18
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.10.2 Abstract test suite for GML of L2.2D.0d

A.10.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.10.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.10.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.10.1
d) Test Type: Capability Test

A.11 Abstract test suite for Profile L2.2D.1d

A.11.1 Abstract test suite for UML of L2.2D.1d

A.11.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.1, 6.3.5, 6.3.13, 6.3.14.1, 6.3.16, 6.4.1, 6.4.6, 6.4.8 - 6.4.31, and 6.5.5.
d) Test Type: Capability Test

A.11.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.10.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.10.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.11.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.4, A.2.6, A.2.8, A.2.9, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107

d) Test Type: Capability Test

A.11.2 Abstract test suite for GML of L2.2D.1d

A.11.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 11.3.2, 11.3.3, 20, 21.12,
d) Test Type: Capability Test

A.11.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.11.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.11.1
d) Test Type: Capability Test

A.12 Abstract test suite for Profile L2.2D.2d

A.12.1 Abstract test suite for UML of L2.2D.2d

A.12.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.3.6, 6.3.7, 6.3.10.4, 6.3.15, 6.3.17.1, 6.3.17.3, 6.4.6, 6.4.32 - 6.4.48, and 6.5.6
d) Test Type: Capability Test

A.12.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.11.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_MultiSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.11.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.12.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.4, A.2.6, A.2.8, A.2.9, A.2.10, A.2.13, A.2.14, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.12.2 Abstract test suite for GML of L2.2D.2d

A.12.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 10.5.1, 10.5.2, 10.5.4 - 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 - 10.5.11.6, 11.3.2, 11.3.3, 11.3.4, 20, 21.12
d) Test Type: Capability Test

A.12.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.12.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.12.1
d) Test Type: Capability Test

A.13 Abstract test suite for Profile L2.3D.0d

A.13.1 Abstract test suite for UML of L2.3D.0d

A.13.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.13.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.13.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

e) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.4, A.2.6, A.2.17, A.2.18
f) Test Method: Inspect the implementation.
g) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
h) Test Type: Capability Test

A.13.2 Abstract test suite for GML of L2.3D.0d

A.13.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.13.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.13.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.13.1
d) Test Type: Capability Test

A.14 Abstract test suite for Profile L2.3D.1d

A.14.1 Abstract test suite for UML of L2.3D.1d

A.14.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.1, 6.3.5, 6.3.13, 6.3.14.1, 6.3.16, 6.4.1, 6.4.6, 6.4.8 - 6.4.31, and 6.5.5.
d) Test Type: Capability Test

A.14.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.13.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.13.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.14.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.4, A.2.6, A.2.8, A.2.9, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.14.2 Abstract test suite for GML of L2.3D.1d

A.14.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 11.3.2, 11.3.3, 20, 21.12
d) Test Type: Capability Test

A.14.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.14.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.14.1
d) Test Type: Capability Test

A.15 Abstract test suite for Profile L2.3D.2d

A.15.1 Abstract test suite for UML of L2.3D.2d

A.15.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.3.6, 6.3.7, 6.3.10.4, 6.3.15, 6.3.17.1, 6.3.17.3, 6.4.6, 6.4.32-6.4.48, and 6.5.6
d) Test Type: Capability Test

A.15.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.14.1.2 in this document and in addition implements ISO 19107 classes: GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_PointGrid, GM_SurfaceInterpolation, GM_SurfacePatch, GM_PolyhedralSurface, GM_Polygon, GM_TriangulatedSurface, GM_Triangle, GM_Tin, GM_ParametricCurveSurface, GM_GriddedSurface, GM_BilinearGrid, GM_MultiSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.14.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.15.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.4, A.2.6, A.2.8, A.2.9, A.2.10, A.2.13, A.2.15, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.15.2 Abstract test suite for GML of L2.3D.2d

A.15.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.

c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 10.5.1, 10.5.2, 10.5.4 - 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 - 10.5.11.6, 11.3.2, 11.3.3, 11.3.4, 20, 21.12
d) Test Type: Capability Test

A.15.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.15.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.15.1
d) Test Type: Capability Test

A.16 Abstract test suite for Profile L2.3D.3d

A.16.1 Abstract test suite for UML of L2.3D.3d

A.16.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.4 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, 6.3.8, 6.3.9, 6.3.10.4, 6.3.18.1 and 6.5.7
d) Test Type: Capability Test

A.16.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.15.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Shell, GM_SolidBoundary, GM_Solid, GM_MultiSolid,
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.15.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.16.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.4, A.2.6, A.2.8, A.2.9, A.2.10, A.2.13, A.2.15, A.2.16, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.16.2 Abstract test suite for GML of L2.3D.3d

A.16.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.4 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, 10.6.1, 10.6.2, 10.6.4 -10.6.6, 11.3.5, 20, 21.12
d) Test Type: Capability Test

A.16.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.16.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.16.1
d) Test Type: Capability Test

A.17 Abstract test suite for Profile L3.2D.0d

A.17.1 Abstract test suite for UML of L3.2D.0d

A.17.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.

c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.17.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position , GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.17.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.17, A.2.18
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.17.2 Abstract test suite for GML of L3.2D.0d

A.17.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.17.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.17.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.17.1
d) Test Type: Capability Test

A.18 Abstract test suite for Profile L3.2D.1d

A.18.1 Abstract test suite for UML of L3.2D.1d

A.18.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.1, 6.3.5, 6.3.13, 6.3.14.1, 6.3.16, 6.4.1, 6.4.6, 6.4.8 - 6.4.31, and 6.5.5.
d) Test Type: Capability Test

A.18.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.17.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.17.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.18.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107

d) Test Type: Capability Test

A.18.2 Abstract test suite for GML of L3.2D.1d

A.18.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 11.3.2, 11.3.3, 20, 21.12,
d) Test Type: Capability Test

A.18.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.18.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.18.1
d) Test Type: Capability Test

A.19 Abstract test suite for Profile L3.2D.2d

A.19.1 Abstract test suite for UML of L3.2D.2d

A.19.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.3.6, 6.3.7, 6.3.10.4, 6.3.15, 6.3.17.1, 6.3.17.3, 6.4.6, 6.4.32 - 6.4.48, and 6.5.6
d) Test Type: Capability Test

A.19.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.18.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_MultiSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.11.1.2 , DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.19.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.12, A.2.13, A.2.14, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.19.2 Abstract test suite for GML of L3.2D.2d

A.19.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 10.5.1, 10.5.2, 10.5.4 - 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 - 10.5.11.6, 11.3.2, 11.3.3, 11.3.4, 20, 21.12
d) Test Type: Capability Test

A.19.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.19.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.19.1
d) Test Type: Capability Test

A.20 Abstract test suite for Profile L3.3D.0d

A.20.1 Abstract test suite for UML of L3.3D.0d

A.20.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference:ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.20.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.20.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.17, A.2.18
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.20.2 Abstract test suite for GML of L3.3D.0d

A.20.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.20.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.20.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.20.1
d) Test Type: Capability Test

A.21 Abstract test suite for Profile L3.3D.1d

A.21.1 Abstract test suite for UML of L3.3D.1d

A.21.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.1, 6.3.5, 6.3.13, 6.3.14.1, 6.3.16, 6.4.1, 6.4.6, 6.4.8 - 6.4.31, and 6.5.5.
d) Test Type: Capability Test

A.21.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.20.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.20.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.21.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.21.2 Abstract test suite for GML of L3.3D.1d

A.21.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 11.3.2, 11.3.3, 20, 21.12,
d) Test Type: Capability Test

A.21.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.21.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.21.1
d) Test Type: Capability Test

A.22 Abstract test suite for Profile L3.3D.2d

A.22.1 Abstract test suite for UML of L3.3D.2d

A.22.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.3.6, 6.3.7, 6.3.10.4, 6.3.15, 6.3.17.1, 6.3.17.3, 6.4.6, 6.4.32 - 6.4.48, and 6.5.6
d) Test Type: Capability Test

A.22.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.21.1.2 in this document and in addition implements ISO 19107 classes: GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_PointGrid, GM_SurfaceInterpolation, GM_SurfacePatch, GM_PolyhedralSurface, GM_Polygon, GM_TriangulatedSurface, GM_Triangle, GM_Tin, GM_ParametricCurveSurface, GM_GriddedSurface, GM_BilinearGrid, GM_MultiSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.21.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.22.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.5, A.2.6, A.2.13, A.2.15, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.22.2 Abstract test suite for GML of L3.3D.2d

A.22.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.

c) Reference: ISO 19136, Clause 10.3, 10.4.1, 10.4.2, 10.4.4, 10.4.5, 10.4.7, 10.4.7.4 - 10.4.7.21, 10.5.1, 10.5.2, 10.5.4 - 10.5.9, 10.5.10, 10.5.11.1, 10.5.12.4 - 10.5.11.6, 11.3.2, 11.3.3, 11.3.4, 20, 21.12
d) Test Type: Capability Test

A.22.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.22.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.22.1
d) Test Type: Capability Test

A.23 Abstract test suite for Profile L3.3D.3d

A.23.1 Abstract test suite for UML of L3.3D.3d

A.23.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.4 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, 6.3.8, 6.3.9, 6.3.10.4, 6.3.18.1 and 6.5.7
d) Test Type: Capability Test

A.23.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.22.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Shell, GM_SolidBoundary, GM_Solid, GM_MultiSolid,
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.22.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.23.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.5, A.2.6, A.2.13, A.2.15, A.2.16, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.23.2 Abstract test suite for GML of L3.3D.3d

A.23.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.4 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, 10.6.1, 10.6.2, 10.6.4 -10.6.6, 11.3.5, 20, 21.12
d) Test Type: Capability Test

A.23.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.23.1in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.23.1
d) Test Type: Capability Test

A.24 Abstract test suite for Profile L4.2D.0d

A.24.1 Abstract test suite for UML of L4.2D.0d

A.24.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.

c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.24.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_Complex, GM_Composite, GM_CompositePoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.24.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.17, A.2.18, A.2.25
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.24.2 Abstract test suite for GML of L4.2D.0d

A.24.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.24.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.24.1in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.24.1
d) Test Type: Capability Test

A.25 Abstract test suite for Profile L4.2D.1d

A.25.1 Abstract test suite for UML of L4.2D.1d

A.25.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.2.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.6.3, 6.6.1, 6.6.2.1, 6.6.2.3, 6.6.2.4, and 6.6.3 - 6.6.5.
d) Test Type: Capability Test

A.25.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.24.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve, GM_CompositeCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.24.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.25.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107

d) Test Type: Capability Test

A.25.2 Abstract test suite for GML of L4.2D.1d

A.25.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.3.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.2, 10.4.6, 11.2.1.1, 11.2.1.2, 11.2.2.1, 11.2.2.2, 20, 21.12
d) Test Type: Capability Test

A.25.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.25.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.25.1
d) Test Type: Capability Test

A.26 Abstract test suite for Profile L4.2D.2d

A.26.1 Abstract test suite for UML of L4.2D.2d

A.26.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.2.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, A.2.1.1, and 6.6.6
d) Test Type: Capability Test

A.26.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.25.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_MultiSurface, GM_CompositeSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.25.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.26.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.12, A.2.13, A.2.14, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.26.2 Abstract test suite for GML of L4.2D.2d

A.26.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.3.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, A.2.3.1.1, 10.5.11, 11.2.2.3, 20, 21.12
d) Test Type: Capability Test

A.26.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.26.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.26.1
d) Test Type: Capability Test

A.27 Abstract test suite for Profile L4.3D.0d

A.27.1 Abstract test suite for UML of L4.3D.0d

A.27.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.27.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_Complex, GM_Composite, GM_CompositePoint
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.27.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.17, A.2.18, A.2.25
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.27.2 Abstract test suite for GML of L4.3D.0d

A.27.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.27.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.27.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.27.1
d) Test Type: Capability Test

A.28 Abstract test suite for Profile L4.3D.1d

A.28.1 Abstract test suite for UML of L4.3D.1d

A.28.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.2.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, 6.6.3, 6.6.1, 6.6.2.1, 6.6.2.3, 6.6.2.4, and 6.6.3 - 6.6.5
d) Test Type: Capability Test

A.28.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.27.1.2 and in addition implements ISO 19107 classes: GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve, GM_CompositeCurve
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.27.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.28.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.6, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.28.2 Abstract test suite for GML of L4.3D.1d

A.28.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.3.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.2, 10.4.6, 11.2.1.1, 11.2.1.2, 11.2.2.1, 11.2.2.2, 20, 21.12,
d) Test Type: Capability Test

A.28.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.28.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.28.1
d) Test Type: Capability Test

A.29 Abstract test suite for Profile L4.3D.2d

A.29.1 Abstract test suite for UML of L4.3D.2d

A.29.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.2.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, A.2.1.1, and 6.6.6
d) Test Type: Capability Test

A.29.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.28.1.2 in this document and in addition implements ISO 19107 classes: GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_PointGrid, GM_SurfaceInterpolation, GM_SurfacePatch, GM_PolyhedralSurface, GM_Polygon, GM_TriangulatedSurface, GM_Triangle, GM_Tin, GM_ParametricCurveSurface, GM_GriddedSurface, GM_BilinearGrid, GM_MultiSurface, GM_CompositeSurface
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.28.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.29.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.5, A.2.6, A.2.13, A.2.15, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24, A.2.25
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.29.2 Abstract test suite for GML of L4.3D.2d

A.29.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.3.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, A.2.3.1.1, 10.5.11, 11.2.2.3, 20, 21.12
d) Test Type: Capability Test

A.29.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.29.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.29.1
d) Test Type: Capability Test

A.30 Abstract test suite for Profile L4.3D.3d

A.30.1 Abstract test suite for UML of L4.3D.3d

A.30.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.2.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.4, A.2.1.2, and 6.6.7
d) Test Type: Capability Test

A.30.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.29.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Shell, GM_SolidBoundary, GM_Solid, GM_MultiSolid, GM_CompositeSolid
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.29.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.30.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.3, A.2.5, A.2.6, A.2.13, A.2.15, A.2.16, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24, A.2.25,
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.30.2 Abstract test suite for GML of L4.3D.3d

A.30.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.3.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.4, A.2.3.1.2, 11.2.2.4, 20, 21.12
d) Test Type: Capability Test

A.30.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.30.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.30.1
d) Test Type: Capability Test

A.31 Abstract test suite for Profile L5.2D.0d

A.31.1 Abstract test suite for UML of L5.2D.0d

A.31.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.31.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position , GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_Complex, GM_Composite, GM_CompositePoint, TP_Object, TP_Boundary, TP_PrimitiveBoundary, TP_Primitive, TP_DirectedTopo, TP_Node, TP_DirectedNode, TP_Complex
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.31.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.17, A.2.18, A.2.25, A.2.26, A.2.27, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.31.2 Abstract test suite for GML of L5.2D.0d

A.31.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.31.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.31.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.31.1
d) Test Type: Capability Test

A.32 Abstract test suite for Profile L5.2D.1d

A.32.1 Abstract test suite for UML of L5.2D.1d

A.32.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, A.3.1.1, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.32.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.31.1.2 and in addition implements ISO 19107 classes: GM_CurveBoundary, GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve, GM_CompositeCurve, TP_EdgeBoundary, TP_Edge, TP_DirectedEdge, TP_Expression
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.31.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.32.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25, A.2.26, A.2.27, A.2.28, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.32.2 Abstract test suite for GML of L5.2D.1d

A.32.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.1, A.2.2.1.2, A.2.4.1.1, 13.3.2, 13.3.3, 20, 21.12
d) Test Type: Capability Test

A.32.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.32.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.32.1
d) Test Type: Capability Test

A.33 Abstract test suite for Profile L5.2D.2d

A.33.1 Abstract test suite for UML of L5.2D.2d

A.33.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, A.3.1.2, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.33.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.32.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface, GM_MultiSurface, GM_CompositeSurface, TP_FaceBoundary, TP_Ring, TP_Face, TP_DirectedFace
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.32.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.33.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.12, A.2.13, A.2.14, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25, A.2.26, A.2.27, A.2.28, A.2.29, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.33.2 Abstract test suite for GML of L5.2D.2d

A.33.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, A.2.4.1.2, 13.3.4, 20, 21.12
d) Test Type: Capability Test

A.33.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.33.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.33.1
d) Test Type: Capability Test

A.34 Abstract test suite for Profile L5.3D.0d

A.34.1 Abstract test suite for UML of L5.3D.0d

A.34.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.1.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, 6.1, 6.2.1, 6.2.2.17, 6.3.10.1, 6.3.11.1, 6.3.11.2, 6.4.1, 6.5.1, 6.5.2.1, 6.5.2.2, 6.5.3 and 6.5.4
d) Test Type: Capability Test

A.34.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements the ISO 19107 classes: GM_Object, GM_Primitive, GM_Point, DirectPosition, GM_Envelope, GM_Position, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_Complex, GM_Composite, GM_CompositePoint, TP_Object, TP_Boundary, TP_PrimitiveBoundary, TP_Primitive, TP_DirectedTopo, TP_Node, TP_DirectedNode, TP_Complex
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, 19107 DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.34.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.17, A.2.18, A.2.25, A.2.26, A.2.27, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.34.2 Abstract test suite for GML of L5.3D.0d

A.34.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.2.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, Clause 10.3, 11.3.2, 20, 21.12
d) Test Type: Capability Test

A.34.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.34.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.34.1
d) Test Type: Capability Test

A.35 Abstract test suite for Profile L5.3D.1d

A.35.1 Abstract test suite for UML of L5.3D.1d

A.35.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.1 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.2, A.3.1.1, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.35.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.34.1.2 and in addition implements ISO 19107 classes: GM_CurveBoundary, GM_OrientablePrimitive, GM_OrientableCurve, GM_Curve, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_MultiCurve, GM_CompositeCurve, TP_EdgeBoundary, TP_DirectedTopo, TP_Edge, TP_DirectedEdge, TP_Expression
b) Test Method: Inspect the implementation.

c) Reference DGIWG GML Profiles, Clause A.34.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.35.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.13, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25, A.2.26, A.2.27 A.2.32, A.2.33, A.2.34,
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.35.2 Abstract test suite for GML of L5.3D.1d

A.35.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.1 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.1, A.2.2.1.2, A.2.4.1.1, 13.3.2, 13.3.3
d) Test Type: Capability Test

A.35.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.35.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.35.1
d) Test Type: Capability Test

A.36 Abstract test suite for Profile L5.3D.2d

A.36.1 Abstract test suite for UML of L5.3D.2d

A.36.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, A.3.1.2, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.36.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.35.1.2 in this document and in addition implements ISO 19107 classes: GM_Boundary, GM_PrimitiveBoundary, GM_Ring, GM_SurfaceBoundary, GM_OrientableSurface, GM_Surface,GM_PointGrid, GM_SurfaceInterpolation, GM_SurfacePatch, GM_PolyhedralSurface, GM_Polygon, GM_TriangulatedSurface, GM_Triangle, GM_Tin, GM_ParametricCurveSurface, GM_GriddedSurface, GM_BilinearGrid, GM_MultiSurface, GM_CompositeSurface, TP_FaceBoundary, TP_Ring, TP_Face, TP_DirectedFace
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.35.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.36.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.12, A.2.13, A.2.15, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24, A.2.25, A.2.26, A.2.27, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.36.2 Abstract test suite for GML of L5.3D.2d

A.36.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, A.2.4.1.2, 13.3.4, 20, 21.12
d) Test Type: Capability Test

A.36.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.36.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.36.1
d) Test Type: Capability Test

A.37 Abstract test suite for Profile L5.3D.3d

A.37.1 Abstract test suite for UML of L5.3D.3d

A.37.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.4, A.3.1.3, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.37.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile conforms to conformance class A.36.1.2 in this document and in addition implements ISO 19107 classes: GM_Shell, GM_SolidBoundary, GM_Solid, GM_MultiSolid, GM_CompositeSolid, TP_SolidBoundary, TP_Shell, TP_Solid, TP_DirectedSolid
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, Clause A.36.1.2, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.37.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.12, A.2.13, A.2.15, A.2.16, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24, A.2.25, A.2.26, A.2.27, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.37.2 Abstract test suite for GML of L5.3D.3d

A.37.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.4, A.2.4.1.3, 13.3.5, 20, 21.12
d) Test Type: Capability Test

A.37.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.37.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.37.1
d) Test Type: Capability Test

A.38 Abstract test suite for Profile L6.2D.2d

A.38.1 Abstract test suite for UML of L6.2D.2d

A.38.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.2 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.3, A.3.1.2, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.38.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements ISO 19107 classes: GM_Object, GM_Boundary, GM_PrimitiveBoundary, GM_CurveBoundary, GM_Ring, GM_SurfaceBoundary, GM_Primitive, GM_Point, GM_OrientablePrimitive, GM_OrientableCurve, GM_OrientableSurface, GM_Curve, GM_Surface, DirectPosition, GM_Envelope, GM_Position, GM_PointArray, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, GM_Complex, GM_Composite, GM_CompositePoint, GM_CompositeCurve, GM_CompositeSurface, TP_Object, TP_Boundary, TP_PrimitiveBoundary, TP_EdgeBoundary, TP_FaceBoundary, TP_Ring, TP_Primitive, TP_DirectedTopo, TP_Node, TP_DirectedNode, TP_Edge, TP_DirectedEdge, TP_Face, TP_DirectedFace, TP_Expression, TP_Complex
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.38.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.12, A.2.13, A.2.14, A.2.17, A.2.18, A.2.19, A.2.20, A.2.25, A.2.26, A.2.27, A.2.28, A.2.30, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.38.2 Abstract test suite for GML of L6.2D.2d

A.38.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.2 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.3, A.2.4.1.2, 13.3.4, 20, 21.12
d) Test Type: Capability Test

A.38.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.38.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.38.1
d) Test Type: Capability Test

A.39 Abstract test suite for Profile L6.3D.3d

A.39.1 Abstract test suite for UML of L6.3D.3d

A.39.1.1 Abstract test for ISO 19107 conformance

a) Test Purpose: Verify that the UML profile conforms to the test A.4.1.3 in the ISO 19107 standard.
b) Test Method: Inspect the implementation.
c) Reference: ISO 19107, A.1.1.4, A.3.1.3, 7.3.8.2 and 7.4.2.8
d) Test Type: Capability Test

A.39.1.2 Abstract test for profile class implementation

a) Test Purpose: Verify that the profile implements ISO 19107 classes: GM_Object, GM_Boundary, GM_PrimitiveBoundary, GM_CurveBoundary, GM_Ring, GM_SurfaceBoundary, GM_Shell, GM_SolidBoundary, GM_Primitive, GM_Point, GM_OrientablePrimitive, GM_OrientableCurve, GM_OrientableSurface, GM_Curve, GM_Surface, GM_Solid, DirectPosition, GM_Envelope, GM_Position, GM_PointArray, GM_PointGrid, GM_CurveInterpolation, GM_CurveSegment, GM_LineString, GM_GeodesicString, GM_ArcString, GM_SurfaceInterpolation, GM_SurfacePatch, GM_PolyhedralSurface, GM_Polygon, GM_TriangulatedSurface, GM_Triangle, GM_Tin, GM_ParametricCurveSurface, GM_GriddedSurface, GM_BilinearGrid, GM_Aggregate, GM_MultiPrimitive, GM_MultiPoint, GM_MultiCurve, GM_MultiSurface, GM_MultiSolid, GM_Complex, GM_Composite, GM_CompositePoint, GM_CompositeCurve, GM_CompositeSurface, GM_CompositeSolid, TP_Object, TP_Boundary, TP_PrimitiveBoundary, TP_EdgeBoundary, TP_FaceBoundary, TP_SolidBoundary, TP_Ring, TP_Shell, TP_Primitive, TP_DirectedTopo, TP_Node, TP_DirectedNode, TP_Edge, TP_DirectedEdge, TP_Face, TP_DirectedFace, TP_Solid, TP_DirectedSolid, TP_Expression, TP_Complex
b) Test Method: Inspect the implementation.
c) Reference DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.39.1.3 Abstract test suite for the DGIWG specializations and constraints of ISO 19107

a) Test Purpose: Verify that the following DGIWG specializations and constraints of ISO 19107 are met: A.2.1, A.2.2, A.2.5, A.2.7, A.2.11, A.2.12, A.2.13, A.2.15, A.2.16, A.2.17, A.2.18, A.2.19, A.2.20, A.2.21, A.2.22, A.2.23, A.2.24, A.2.25, A.2.26, A.2.27, A.2.31, A.2.32, A.2.33, A.2.34
b) Test Method: Inspect the implementation.
c) Reference: DGIWG GML Profiles, DGIWG Profiles Matrix for ISO 19107
d) Test Type: Capability Test

A.39.2 Abstract test suite for GML of L6.3D.3d

A.39.2.1 Abstract test for ISO 19136 conformance

a) Test Purpose: Verify that a GML profile conforms to the A.2.1 and A.2.5.1.3 in the ISO 19136 GML standard.
b) Test Method: Inspect the profile schema.
c) Reference: ISO 19136, A.2.2.1.4, A.2.4.1.3, 13.3.5, 20, 21.12
d) Test Type: Capability Test

A.39.2.2 Abstract test for DGIWG ISO 19107 profile conformance

a) Test Purpose: Verify that a GML profile conforms to the tests in Clause A.39.1 in this document
b) Test Method: Inspect the profile schema.
c) Reference: DGIWG GML Profiles, Clause A.39.1
d) Test Type: Capability Test

Annex B – Abstract test suite for GML application schema

B.1 Introduction

This Annex provides abstract test suites for application schemas build upon the DGIWG GML profiles. These abstract tests are hierarchical such that each test includes all tests of the preceding level (for example, test for L3_2D conformance levels includes tests from L1_2D and L2_2D).

Each application profile implementing one of the twelve compliance level shall first conforme to B.2.

B.2 General test for all GML application schema : import the compliance levels schema

a) Test Purpose: Verify that the GML application schema imports the DGIWG conformance level GML schema (gmlDGIWGspLevels.xsd from the DGIWG namespace) as required in 19.2.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3 Tests for 2D spatial dimension

B.3.1 Conformance level L1_2D

B.3.1.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the followings geometric and topologic GML classes: gml:Point, gml:PointPropertyType, gml:MultiPoint, gml:MultiPointPropertyType, gml:Curve, gml:LineStringSegment, gml:LineString, gml:CurvePropertyType, gml:MultiCurve, gml:MultiCurvePropertyType, gml:Surface, gml:PolygonPatch, gml:Polygon, gml:SurfacePropertyType, gml:LinearRing, gml:Ring, gml:MultiSurface, gml:MultiSurfacePropertyType
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.1.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L1_2D with gml2dGeometry.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.2 Conformance level L2_2D

B.3.2.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.2.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L2_2D with gml2dGeometry.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document

d) Test Type: Capability Test

B.3.3 Conformance level L3_2D

B.3.3.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.3.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L3_2D with gml2dGeometry.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.4 Conformance level L4_2D

B.3.4.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1, plus the following ones : gml:CompositeCurve, gml:OrientableCurve, gml:GeometricComplex, gml:GeometricComplexPropertyType, gml:CompositeSurface, gml:OrientableSurface.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.4.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L4_2D with gml2dComplex.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.5 Conformance level L5_2D

B.3.5.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1, plus the following ones :. gml:TopoComplex, gml:TopoComplexPropertyType, gml:Node, gml:directedNode, gml:Edge, gml:directedEdge, gml:Face, gml:directedFace.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.5.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L5_2D with gml2dTopology.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.3.6 Conformance level L6_2D

B.3.6.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1.

b) Reference: This document
c) Test Type: Capability Test

B.3.6.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L6_2D with gml2dTopology.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4 Tests for 3D spatial dimension

B.4.1 Conformance level L1_3D

B.4.1.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1, plus the following ones: gml:Solid, gml:SolidPropertyType, gml:Shell, gml:MultiSolid, gml:MultiSolidPropertyType.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.1.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L1_3D with gml3dGeometry.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.2 Conformance level L2_3D

B.4.2.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.2.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L2_3D with gml3dGeometry.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.3 Conformance level L3_2D

B.4.3.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.3.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L3_3D with gml3dGeometry.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.

b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.4 Conformance level L4_3D

B.4.4.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.3.1.1, plus the following one : gml:CompositeSolid.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.4.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L4_3D with gml3dComplex.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.5 Conformance level L5_3D

B.4.5.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.4.4.1, plus the following ones : gml:TopoSolid, gml:DirectedTopoSolidPropertyType.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.5.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L5_3D with gml3dTopology.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.6 Conformance level L6_3D

B.4.6.1 GML geometric and topologic classes

a) Test Purpose: Verify that the GML application schema only uses the geometric and topologic GML classes listed in B.4.5.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

B.4.6.2 Declaration in the GML application schema

a) Test Purpose: Verify that an XML annotation is used for indentifying the compliance level of the schema (L6_3D with gml3dTopology.xsd schema) and is placed at the top most nesting level of the schema as required in 9.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

Annex C – Abstract test suites for GML data

C.1 Introduction

This Annex provides abstract test suites for GML data build upon the DGIWG GML profiles. These abstract tests are hierarchical such that each test includes tests of the preceding level (for example, test for L3_2D conformance levels includes tests from L1_2D and L2_2D).

GML data implementing one of the twelve compliance level shall first conform to B.2 .

C.2 General test for GML data

a) Test Purpose: Verify that GML data satisfy requirement in section 10.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

C.3 Tests for 2D & 3D spatial dimension GML data

C.3.1 Conformance level L1_2D and L1_3D

a) Test Purpose: Verify that GML data satisfy requirement in section 10.1.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

C.3.2 Conformance level L2_2D and L2_3D

a) Test Purpose: Verify that GML data satisfy requirement in section 10.2.2.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

C.3.3 Conformance level L3_2D and L3_3D

a) Test Purpose: Verify that GML data satisfy requirement in section 10.2.3.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

C.3.4 Conformance level L4_2D and L4_3D

a) Test Purpose: Verify that GML data satisfy requirement in section 10.2.4.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

C.3.5 Conformance level L5_2D and L5_3D

a) Test Purpose: Verify that GML data satisfy requirement in section 10.2.5.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

C.3.6 Conformance level L6_2D and L6_3D

a) Test Purpose: Verify that GML data satisfy requirement in section 10.2.6.
b) Test Method: Inspect the application schema.
c) Reference: This document
d) Test Type: Capability Test

Annex D – UML Profiling Rules

(informative)

D.1 Introduction

The profiling rules presented in this annex are the collection of rules used to derive the UML profiles in this document. The majority of these rules are adapted from the profiling rules listed in clause B.3 of ISO 19106. Although these rules are not normative they provide a clear example of what is acceptable. Also considered in developing these rules is clause 5.1.5 on strong substitutability of ISO 19107. The significant portions of these two clauses are reproduced here for reader convenience.

From ISO 19106 clause B.3:

> The principle for developing the profile is that the profile shall be a specialization and as such it shall follow certain rules:
>
> — Only classes from ISO 19107 can be used.
>
> — In an inheritance tree certain subclasses can be picked and others can therefore be excluded from the model.
>
> — All attributes and relations (also inherited ones) from ISO 19107 shall be present in the chosen classes.
>
> — Attributes and relations can be specialized.
>
> — A relation at the supertype level can be specialized to the subtype level.
>
> — Multiplicity constraints can be constrained further. I.e. a 0..* can be constrained to 0, 0..1, 1 etc.
>
> — A *set* can be specialized to a *sequence of unique*.
>
> — A *bag* can be specialized to a *set*.
>
> — Constraints can be specialized.
>
> The fundamental idea behind specialization is that the specialization shall fit into the more generalized definition and present no problems for clients of data and/or interfaces that uses the generalized definition. This also follows the rules of substitutability which means that a general class can always be substituted by a specialized class in all contexts.

From ISO 19107 clause 5.1.5:

> This International Standard assumes that implementation profiles and transfer schemas will be built using a strong version of substitutability. This means that at several places in designing an application schema, a profile builder may use a class in lieu of one defined in this schema as long as it supports the data, operation and associations required of the base class. The method of implementation of this substitutability is not normative, and may be done in a variety of manners depending on the characteristics of the implementation environment. This is especially true of transfer standards, which by their nature depend on data types. Entities in transfer sets may only be tenuously related to the base class in this International Standard, in that they may be data-only representational forms.

The following is the list of profiling rules. Each rule is described in more detail, with examples, in its own sub-clause.

1. Subclasses may be omitted.
2. Classes may be made abstract.
3. Multiplicities may be constrained.
4. Operations may be omitted.
5. Operations with no parameters may be changed to attributes or associations.
6. Relations may be specialized to the subtype level.
7. Collections may add structure.
    a. A Bag may become a Set or a Sequence.
    b. A Set may become a Sequence of Unique.
8. Code lists may be subsetted.
9. Constraints may be specialized.
10. Weak aggregations may be made into strong aggregations.
11. Inheritance chains may be flattened (data only profiles).

D.2 Explanation of profiling rules

D.2.1 Subclasses may be omitted

This rule is derived directly from the rules defined in ISO 19106. There the rule is stated as:

—	In an inheritance tree certain subclasses can be picked and others can therefore be excluded from the model.

Given an inheritance tree not all subclasses need to be included in the profile. For example the inheritance tree shown in Figure D.1 could be profiled as shown in Figure D.2.

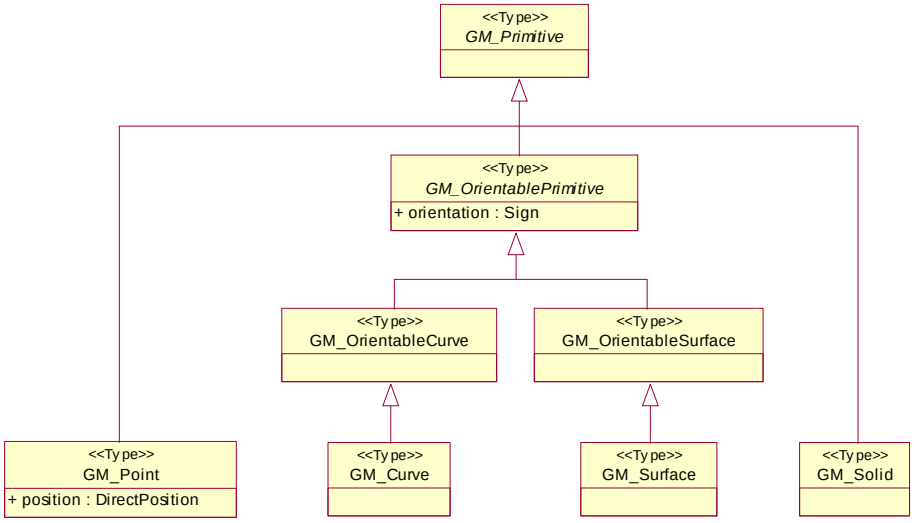


Figure D.1 – ISO 19107 inheritance tree

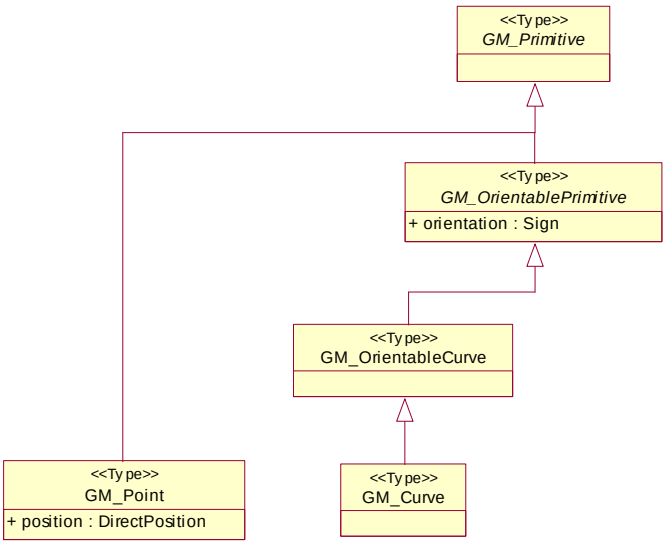


Figure D.2 – Profiled inheritance tree

D.2.2 Classes may be made abstract

In the previous rule subclasses were made uninstantiable by omitting them from the profile. It may be necessary to make parent classes uninstantiable. This may be achieved not by omitting the class but by making the class abstract. That is to say, a parent class, that in the standard is concrete, may be made abstract in the profile. This satisfies the condition of substitutability in that where ever a given class is expected the instances will be of derived classes but never direct instances of the class.

Figure D.3 and Figure D.4 show how GM_Aggregate is concrete in the standard and changed to abstract in the profile. The standard allows GM_Aggregate to be instantiated but the profile only allows aggregates of one kind of primitive.

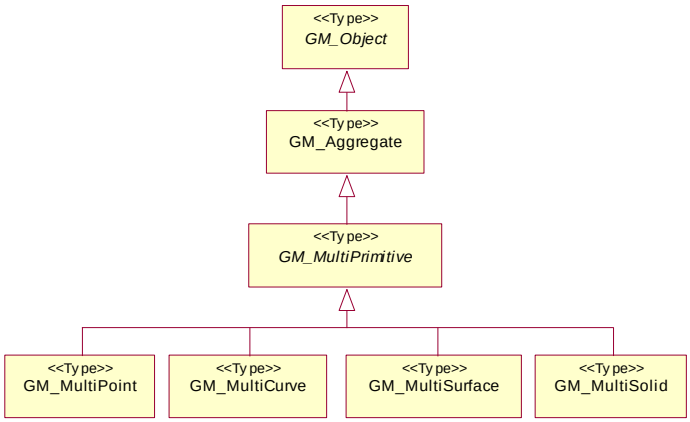


Figure D.3 – ISO 19107 GM_Aggregate inheritance tree

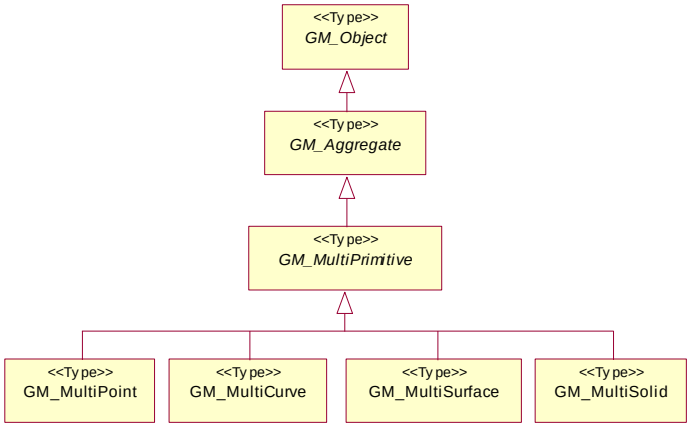


Figure D.4 – Profile GM_Aggregate inheritance tree with GM_Aggregate abstract

D.2.3 Multiplicities may be constrained

This rule is derived directly from the rules define in ISO 19106. There the rule is stated as:

—	Multiplicity constraints can be constrained further. I.e. a 0..* can be constrained to 0, 0..1, 1 etc.

Figure D.5 and Figure D.6 show how the multiplicities of the Contains association are constrained from 0..n to 0. Note that this effectively removes the association from the profile.

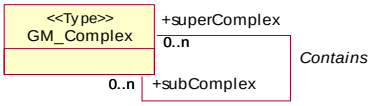


Figure D.5 – ISO 19107 GM_Complex Contains association

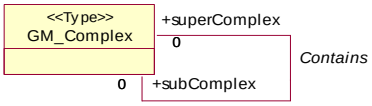


Figure D.6 – Profile GM_Complex Contains association with constrained multiplicity

D.2.4 Operations may be omitted

This rule is a variation on the following rule defined in ISO 19106.

—	All attributes and relations (also inherited ones) from ISO 19107 shall be present in the chosen classes.

The rule states that all attributes and relations shall be present. This implies that operations may be omitted. This is also consistent with the conformance classes of ISO 19107 as well as the fact that the profiles presented in this document are data only profiles.

Figure D.7 shows GM_Object as it is defined in ISO 19107. Figure D.8 shows a profiled version of GM_Object with some but not all operations omitted. Figure D.9 shows a data only profile of GM_Object omitting all operations.

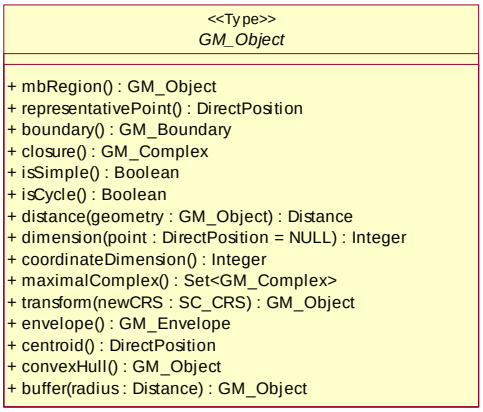


Figure D.7 – Complete GM_Object as defined in ISO 19107

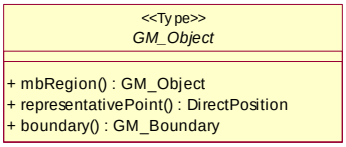


Figure D.8 – Profile GM_Object with some operations

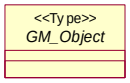


Figure D.9 – Profile GM_Object with no operations

D.2.5 Operations with no parameters may be changed to attributes or associations

Although not immediately apparent, there are a number of reasons why this is a valid transformation of an operation. To begin with, ISO 19107 is a conceptual schema built with types and interfaces. As such it defines behaviour and not implementation. As long as behaviour is retained it is insignificant how it is expressed. The purpose of attributes and associations in types/interfaces is to define state and to specify behaviour, not to determine implementation. This interpretation can be found in both first and second editions of "The Unified Modeling Language Reference Manual" by James Rumbaugh, Ivar Jacobson, and Grady Booch.

In the first edition, chapter 13 Encyclopedia of Terms, under the entry for "type" it states:

> [A type] may also include attributes and associations in order to specify the behavior of operations. A type's attributes and associations do not determine the implementation of its objects.

In the second edition, chapter 14 Dictionary of Terms, under the entry for "interface" it states:

> An interface may have the following structure:
>
> attributes — State values to be maintained by the implementation classifier, not necessarily as attribute values.
>
> operations — Services that unspecified, anonymous objects can invoke of an implementation object. An interface may not provide methods to implement its operations.
>
> :
> :
>
> An interface may have associations to other interfaces. This means that a conforming association must exist between instances of classifiers that realize the interfaces.

Note that between the first and second editions the concept of "type" appears to have become very similar to if not the same as that of "interface".

In a data interchange profile the attributes are read only. An operation with a return value taking no parameters behaves the same as a read only attribute, the two can be used interchangeably. This duality is reflected in programming languages like Eiffel, where a parameterless operation and an attribute are accessed in the same way, and changing from one to the other requires no change in the code that uses them.

The ability to change parameterless operations to attributes or associations is also a matter of necessity. Types that have only operations, and interfaces, which have no attributes, would therefore be empty in data only interchange profiles.

Figure D.10 and Figure D.11 show how the type GM_Solid's parameterless operations are changed to attributes and an association. The modified type makes sense for a data interchange profile where the original representation did not.

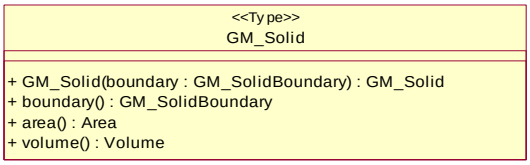


Figure D.10 – GM_Solid as defined in ISO 19107

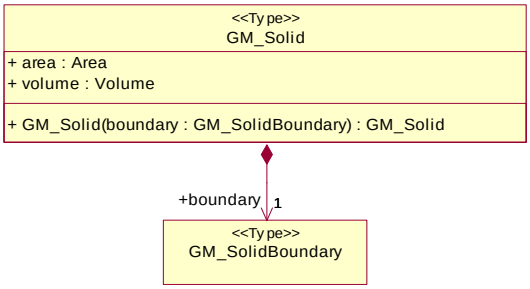


Figure D.11 – GM_Solid in profile. Operations changed to attributes and associations

D.2.6 Relations may be specialized to the subtype level

This rule is derived directly from the rules defined in ISO 19106. There the rule is stated as:

— A relation at the supertype level can be specialized to the subtype level.

This is called covariance in C++ and conformance in Eiffel. ISO 19107 includes a number of examples of this mechanism. Figure D.12 shows how when GM_PolyhedralSurface is specialized to GM_TriangulatedSurface the Segmentation association is specialized as well to reference GM_Triangle which is a specialization of GM_Polygon.

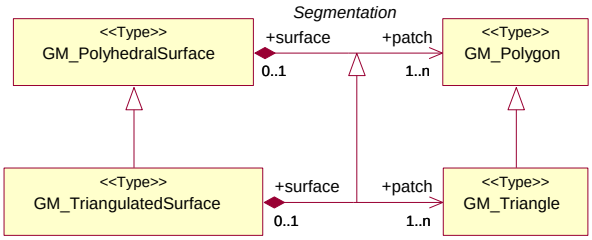


Figure D.12 – Conformance of Segmentation association in ISO 19107

In this example, the Segmentation association is defined as:

```
GM_PolyhedralSurface::patch[1..n] : GM_Polygon
GM_Polygon::surface[0..1] : GM_PolyhedralSurface
```

The association between the two subclasses is conformant to the association between the parent classes:

```
GM_TriangulatedSurface::patch[1..n] : GM_Triangle
```

```
GM_Triangle::surface[0..1] : GM_TriangulatedSurface
```

D.2.7 Collections may add structure

This rule is derived from the rules defined in ISO 19106. It is a generalization of the two rules:

— A *set* can be specialized to a *sequence of unique*.
— A *bag* can be specialized to a *set*.

In the first rule the set becomes a sequence of unique which orders the set in a sequence. A sequence of unique is a kind of set. In the second rule the bag becomes a set which requires every element to be unique. A set is a kind of bag.

The profiling rule used in this document is more general in that any specialization of a collection is valid. For example, in ISO 19107, the TP_DirectedEdges around a TP_Node are collected in a bag, any order with repetition is possible (Figure D.13). In a 2D profile, the bag becomes a circular sequence, order is significant but repetition is possible (Figure D.14).

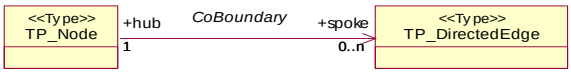


Figure D.13 – ISO 19107 Node/Edge CoBoundary association

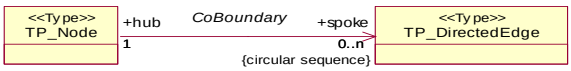


Figure D.14 – Circular Sequence of edges around node in 2D profile

D.2.8 Code lists may be subsetted

This rule follows the condition of substitutability in that anywhere where a value of the original code list is expected a value of the subsetted code list is valid. Figure D.15 and Figure D.16 show how the code list GM_SurfaceInterpolation from ISO 19107 can be subsetted in a profile.

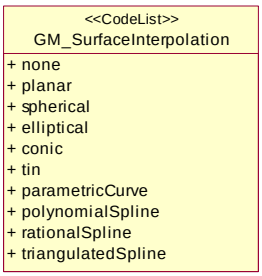


Figure D.15 – ISO 19107 GM_SurfaceInterpolation code list

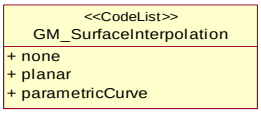


Figure D.16 – Subset of GM_SurfaceInterpolation in profile

D.2.9 Constraints may be specialized

This rule is derived directly from the rules defined in ISO 19106. There the rule is stated as:

— Constraints can be specialized.

The constraint in the profile defines a subset of the constraint in the standard. For example the constraint in ISO 19107

```
context GM_Primitive inv:

       self.dimension() >= coincidentSubelement.dimension()
```

may be further restricted in the profile to

```
context GM_Primitive inv:
        self.dimension() >= coincidentSubelement.dimension() - 2
```

D.2.10 Weak aggregations may be made into strong aggregations

This rule states that a component class that is loosely part of another class may be made to be exclusively part of the containing class. This rule follows the condition of substitutability in that a strong aggregation (composition) will always satisfy the conditions of a weak aggregation. Being a component of zero or one object conforms to being aggregated by zero to many. For example in ISO 19107 a GM_Point can be shared as the boundary of multiple GM_Curves (Figure D.17). In the profile the GM_Point can be made to belong exclusively to a GM_Curve (Figure D.18).

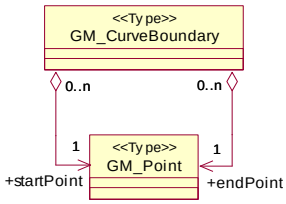


Figure D.17 – ISO 19107 GM_CurveBoundary weak aggregations

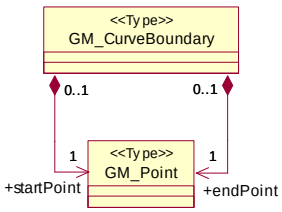


Figure D.18 – Strong aggregations in profile

D.2.11 Inheritance chains may be flattened

In an interchange format a data object contains the attributes and associations of all its parent classes as well as its own. The interchange format flattens the inheritance hierarchy by design. It follows that already flattening inheritance hierarchies in the profile has no affect on the end result but can make the model simpler and therefore clearer.

For example, in ISO 19107 the type GM_Ring is part of an extensive type hierarchy (Figure D.19). In a data only profile this hierarchy can be simplified so that GM_Ring contains all the attributes and associations of its parent classes (Figure D.20).

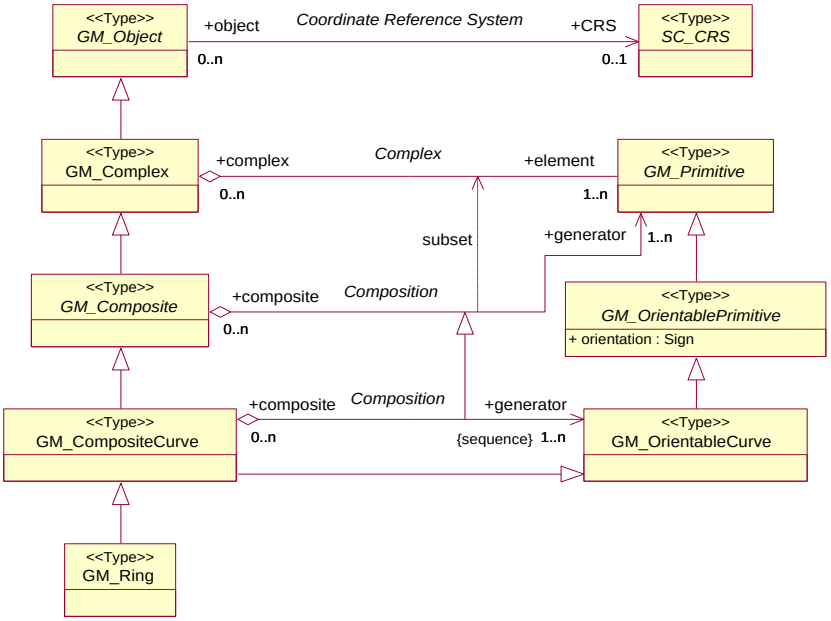


Figure D.19 – Parent classes of GM_Ring in ISO 19107

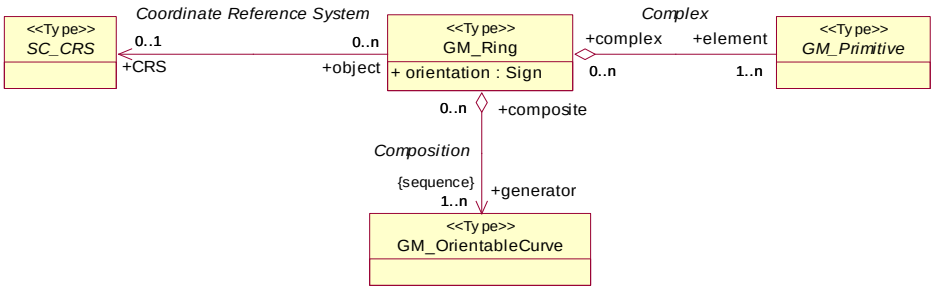


Figure D.20 – Associations and attributes flattened into GM_Ring

Annex E – PERL Mapping script

```
# This script allows for the mapping of UML elements of the different DGIWG Profiles of ISO 19107 to their corresponding GML elements.

# These GML elements can then be used as input parameters to generate GML profiles via the gmlsubset tool provided by the GML standard.

# The script asks the following input parameters

#       - the classfile of the UML elements as a text document

#       - the mapping table as a text document

#       - the name of the output file


$num = $#ARGV;


die "usage: $usage\n" if $#ARGV !=2;


open (CLASSFILE, "<$ARGV[0]") || die ("can't open classfile: $!");


open (MAPPINGTABLE, "<$ARGV[1]") || die ("can't open mappingfile: $!");


open (OUTFILE, ">$ARGV[2]") or die "Can't open output.txt: $!";


@text = <MAPPINGTABLE>;


for $line (<CLASSFILE>)


        {


                @var = ($line =~ /(\w+)/g);


                $such = $var[0];


                @erg = grep (/$such /, @text);


                foreach $erg (@erg)
                        {
                @map = ($erg =~ /([a-zA-Z0-9_:-]+)/g);
```

```
        print "$map[0]\n";

        if ($map[1] !~ /-/)
                {
                        print " $map[1]\n";
                        print OUTFILE "$map[1]\n";

                }

        elsif ($map[2] !~ /-/)
                        {
                                print " $map[2]\n";
                                print OUTFILE "$map[2]\n";
                        }

        elsif ($map[3] !~ /-/)
                        {
                                print " $map[3]\n";
                                print OUTFILE "$map[3]\n";
                        }
        }

}
```

Annex F – Metadata

(informative)

ISO 19136:2007 defines the following rules for referencing GML profiles from application schemas:

“A GML application schema shall reference the full GML schema in the schemaLocation attribute of the <import> element. A GML application schema document conforming to one or more GML Profiles shall provide an appInfo annotation element <gml:gmlProfileSchema> for every profile in the root schema document <schema> element where the value is a schema location of the profile schema. Note that an application schema may conform to multiple profiles.”

The following example was defined according to these rules.

EXAMPLE

```
<schema ...>
    <annotation>
        <appInfo>
            <gml:gmlProfileSchema>http://schemas.dgiwg.org/gmlspatialprofiles/1.0.0/gml2dGeom
                etry.xsd</gml:gmlProfileSchema>
        </appInfo>
    </annotation>
    ...
</schema>
```

The <gml:gmlProfileSchema> element is defined as

```
<element name="gmlProfileSchema" type="anyURI"/>
```

Annex G – Disjoint

(informative)

In this standard the term *disjoint* is used to describe the disjointedness of geometric primitives and refers to the condition where “no direct position is interior to more than one geometric primitive.” In this context it refers exclusively to geometric primitives but is often misinterpreted as referring to geographic features. The disjointedness conditions apply to geometric primitives, the components of geographic area features. The features, in turn, are not necessarily disjoint. Figure G.1 illustrates such an instance where features overlap but the underlying primitives are disjoint. This figure is a satellite image with overlaid colour classification. The orange area is private land, the green area is public land, and the blue overlay is a wetland that extends from the public land through the private land to public land again.

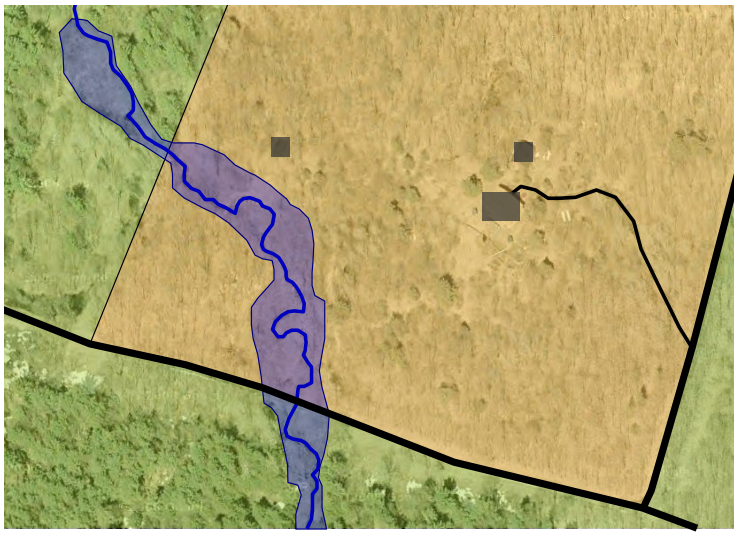


Figure G.1 – Classified image

Figure G.2 removes the underlying image from Figure G.1. In this figure a surface, designated with the letter “S”, is highlighted. This surface is an example of such a shared primitive being shared by, i.e. part of, more than one primitive. This particular surface is shared by both the private land feature and the wetland feature.

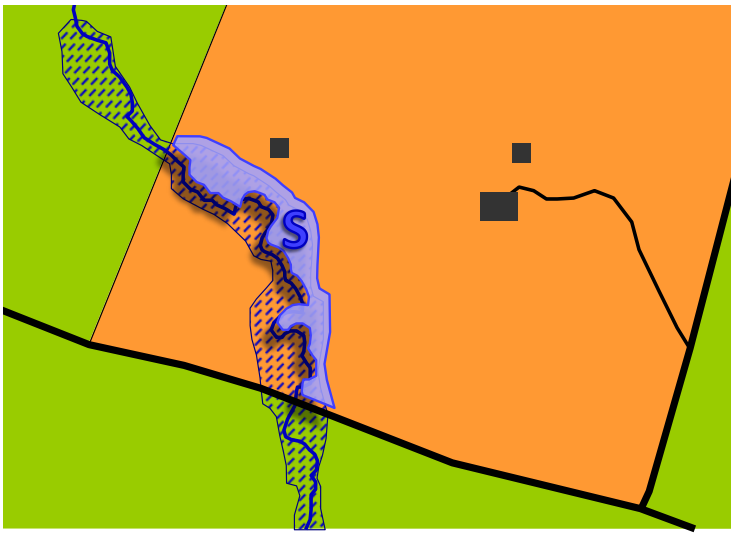


Figure G.2 – An area primitive shared between features

Figure G.3 shows a simplified diagrammatic version of primitive sharing similar to that in Figure G.2. The upper part of the diagram (orange) represents the private land. The lower part of the diagram (green) represents the public land. The blue hatched area is wetland. The wetland extends from the public land into the private land. The point and curve primitives are colour coded, and the surfaces are labelled for reference by Figure G.4.

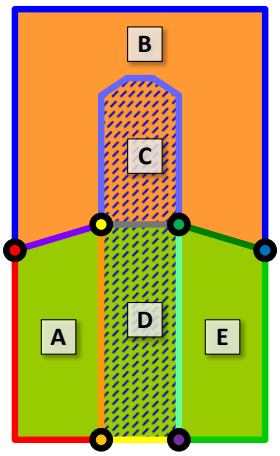


Figure G.3 – Diagrammatic representation of shared primitives

Figure G.4 shows the boundary/coboundary associations between primitives and the aggregation association between features and surface geometry primitives. In particular it shows how the surfaces *C* and *D* are shared between the wetland and, in the case of *C*, the private land and, in the case of *D*, the public land. It is important to note that the surfaces are disjoint while the features are not.

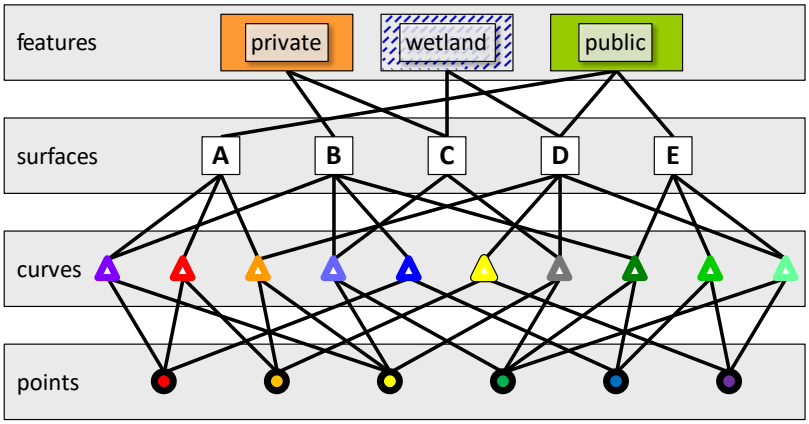


Figure G.4 – Associations between primitives and features

Annex H – What is 2½D?

H.1 Definition

A 2½D profile of ISO 19107 has a 3D spatial dimension, but in all other respects fully conforms to a specific 2D profile definition, and does NOT conform to the corresponding 3D profile definition. As such, a 2½D profile is a 2D profile with 3D coordinates.

There are several rules that distinguish 2½D profiles from 3D profiles:

1) The cobounding curves of a point can be ordered around the point. This rule applies at Levels 4 through 6 when 1D primitives are present.
2) No more than two cobounding surfaces can be associated with any curve. This rule applies at Levels 3 through 6 when 2D primitives are present.
3) Primitives are disjoint in 2D space (i.e. when the third coordinate is dropped). This rule can support stacked Z values (H.2). This rule applies at Levels 3 through 6, and with 0D, 1D, and 2D primitives.

spatial dimension	2D			3D			
primitive dimension	0d	1d	2d	0d	1d	2d	3d
L1 – free geometry	L1.2D.0d	L1.2D.1d	surface completely defined by boundary	L1.3D.0d	L1.3D.1d	L1.3D.2d	L1.3D.3d → 3DFreeGeometry
L2 – no self intersection	L2.2D.0d = L1.2D.0d	L2.2D.1d		L2.3D.0d = L1.2D.0d	L2.3D.1d	L2.3D.2d	L2.3D.3d → 3DSimple
L3 – primitives disjoint	L3.2D.0d (~ L1.2D.0d)	L3.2D.1d		L3.3D.0d (~ L1.2D.0d)	L3.3D.1d	L3.3D.2d	L3.3D.3d → 3DShared
L4 – complex	L4.2D.0d No colocation in 2D	L4.2D.1d orderable cobounding curves	<=2 cobounding surfaces	L4.3D.0d	L4.3D.1d	L4.3D.2d	L4.3D.3d → 3DComplex
L5 – full topology	L5.2D.0d			L5.3D.0d	L5.3D.1d	L5.3D.2d	L5.3D.3d → 3DTopology
L6 – partitioned space							L6.3D.3d → 3DTessellation

fully documented (and named) profiles
described sub-profiles (with constraints)

H.2 Z-bust

Regarding the stacking of Z values in 2½D there are two views, one more strict understanding, and the other a looser interpretation.

Strict 2½D: Strictly interpreted, 2½D is 2D geometry in a 3D coordinate space constrained to a 2D manifold within the 3D space, where, for any X/Y, there is exactly one Z value. Data structured in this manner lies in a single continuous “wrinkled” surface. This type of data is easily rendered by a 3D rendering engine, since there are no singularities in the data. It, on the other hand, misrepresents Z values where the geometry of features intersect in 2D but not in 3D, such as overpasses and bridges. In strict 2½D a bridge and the stream that passes under it necessarily have the same Z value, either that of the bridge, that of the stream, or somewhere in between.

Loose 2½D: Loosely interpreted, 2½D is 2D geometry in a 3D coordinate space where Z values are attributes of vertices and do not factor in collocation and disjointedness conditions. Here the 2D manifold exists within X/Y only. Data structured in this manner is consistent 2D data in X/Y but may have singularities in Z. This type of data more accurately represents the Z values of feature geometries than strict 2½D does, as with overpasses and bridges. In loose 2½D a bridge does pass over the stream that flows under it. Curves representing the stream and the bridge are broken in data where primitives are required to be disjoint. The disjointedness condition is tested in X/Y and the curves are broken at the X/Y intersection. The Z values are unaffected by this and remain at their original or interpolated values. The Z value of the bridge at the intersection remains at its original position above the Z value of the stream at the intersection. This type of data may not render correctly in 3D rendering engines because of the possible singularities in the data, and in this way is closer to 2D data than to 3D. On the other hand, by adding curves and multiple points at stacked Z locations it is possible to convert loose 2½D data to 3D data. This is not possible with strict 2½D data since the true Z values are lost because of the unique Z constraint. Unlike strict 2½D data, loose 2½D data supports network analysis. And, since the Z values are not constrained in loose 2½D data, by constraining the Z value, it can be easily converted to strict 2½D data, where as the reverse is not possible.

Annex I – Complexes

(informative)

I.1 Introduction

This annex clarifies the concept of "complex" and differentiates it from the types GM_Complex and TP_Complex. In this annex the focus is on geometric complexes as opposed to topological complexes. As far as this document is concerned a topological complex corresponds 1-to-1 to a geometric complex.

I.2 Complex

As defined in ISO 19107 a geometric complex is a set of disjoint geometric primitives where the boundary of each geometric primitive can be represented as the union of other geometric primitives of smaller dimension within the same set. As a first step in understanding complexes it is important to understand the concepts of *interior*, *boundary*, and *closure*. A geometric object can be decomposed into interior and boundary. The interior is the set of all direct positions that are on the geometric object but which are not on its boundary, where the boundary is the set that represents the limit of the geometric object. The closure is the union of the interior and the boundary of the geometric object (Figure I.1 and Figure I.2). Note in Figure I.2 that the curve with empty points is open and the curve with solid points is closed.

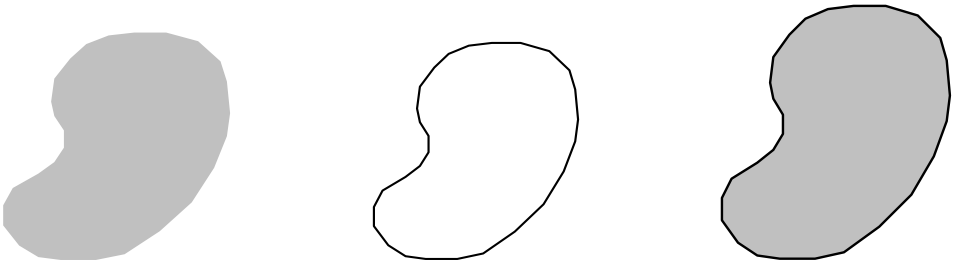


Figure I.1 – Surface interior, boundary, and closure

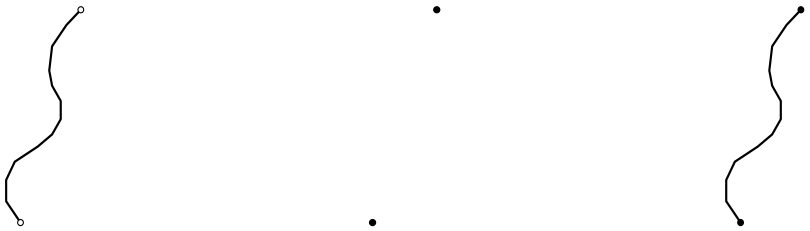


Figure I.2 – Curve interior, boundary, and closure

Figure I.3 shows an open geometric object and its closure.

Figure I.3 – Surface open and closed

Using the concept of closure ISO 19107 goes on to explain that the set of geometric primitives is closed under boundary operations, meaning that for each element in the geometric complex, there is a collection (also a geometric complex) of geometric primitives that represents the boundary of that element.

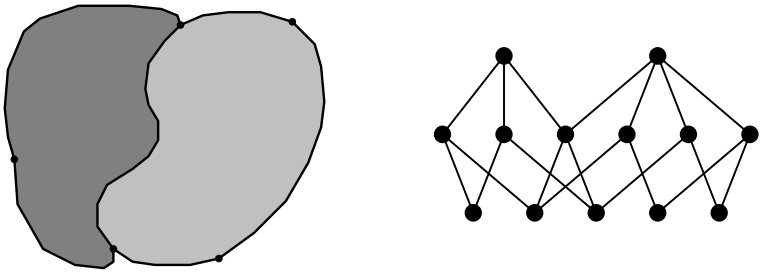


Figure I.4 – Example complex with primitive hierarchy

From the previous statement it is apparent that complexes may have subcomplexes. In fact all complexes of 1-dimension or greater have subcomplexes. Figure I.4 shows an example complex and its corresponding primitive hierarchy. In this hierarchy the surfaces are in the top row, the curves in the middle, and the points in the bottom. The connections between the nodes in the hierarchy signify the boundary relationships. The example complex in Figure I.4 has a number of subcomplexes.

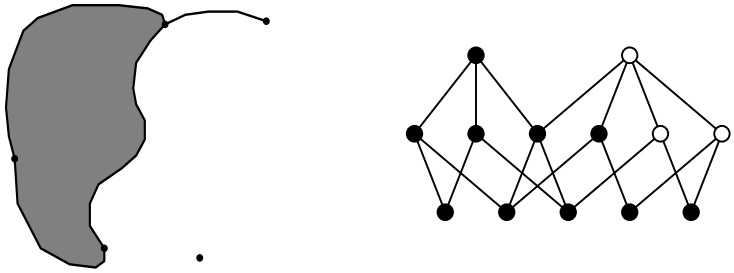


Figure I.5 – Example subcomplex with primitive hierarchy

Figure I.5 shows an example of such a subcomplex. The solid nodes in the hierarchy show primitives that are part of the subcomplex while the empty nodes are not. Of the 8192 (= 2^{13}) possible primitive combinations in this simple example 210 are complexes. The fact that a collection of primitives is not a complex does not indicate that the primitives are unbounded. The primitives may very well be bounded, and individually may be complexes, but as a collection the primitives may not be disjoint. On the other hand a collection may have disjoint primitives where most but not all primitives are bounded. Such a collection is not a complex, although subsets of the collection may be complexes.

I.3 Realization of GM_Complex

The Geometry package of ISO 19107 defines the type GM_Complex that models the structure and behaviour of a complex. As is evident from the example in Figure I.4 above, it makes little sense to create an instance of GM_Complex for every one of the 210 possible complexes. Classes should realize GM_Complex where the interface this type offers serves a purpose. If the purpose is for a dataset to be a complex then only the dataset needs to realize GM_Complex (Figure I.6). If, on the other hand, the behaviour of a complex is required also for features and primitives then they need to realize GM_Complex as well (Figure I.7).

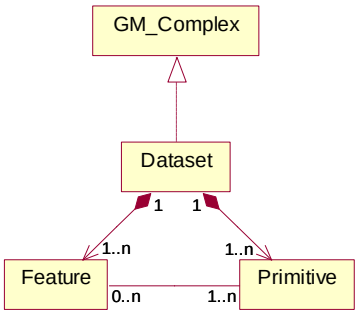


Figure I.6 – Dataset realizing GM_Complex

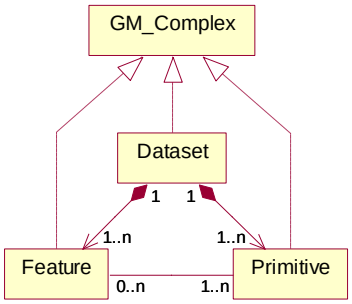


Figure I.7 – Dataset and dataset elements realizing GM_Complex

Annex J – InteriorTo and IsolatedIn associations

(informative)

J.1 Purpose

This annex clarifies and differentiates the associations InteriorTo of the Geometry package and IsolatedIn of the Topology package.

J.2 InteriorTo

As stated in ISO 19107 the InteriorTo association associates GM_Primitives which are coincident with one another. Figure J.1 shows an example curve and an example surface. Each primitive is a bounded but infinite set of direct positions, that is a geometric set as defined in ISO 19107. Subsets of these sets also form geometric primitives. These subset primitives are interior to their superset primitives (Figure J.2). The interior primitives do not pierce their containing primitives but lie on top of them, so to speak. The interior primitives are of the same dimension as the containing primitive or less. That is points and curves can be interior to curves; points, curves, and surfaces can be interior to surfaces; and points, curves, surfaces, and solids can be interior to solids. This interpretation is strictly a non-complex interpretation since in a complex no two primitives may be coincident.

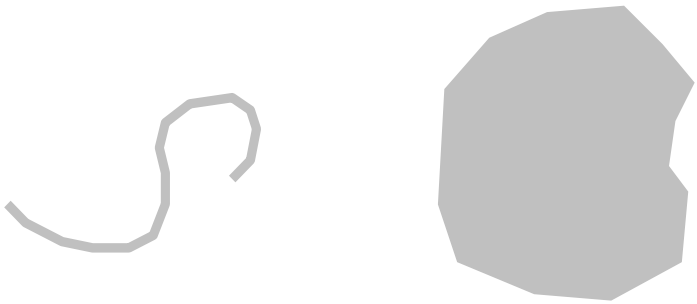


Figure J.1 – Example curve and surface

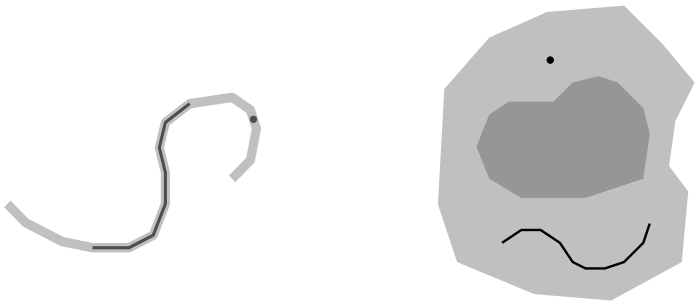


Figure J.2 – Example curve and surface with interior primitives

ISO 19107 states that in a complex the boundary information is sufficient to represent adjacency relations in most cases. The cases where the boundary information is not sufficient are those where the difference in dimension between the primitives is greater than 1. In the case of a complex the semantics of the interior-to association is different. In this case the primitives are no longer coincident but adjacent to each other. Within a complex any given direct location is an element of only one primitive. For example, if a curve lies within a surface of a complex the direct positions that make up the curve are not part of the surface.

The profiles in this document only use this second interpretation of the InteriorTo association.

J.3 IsolatedIn

In the Topology Package of ISO 19107 primitives are required to be in a complex. As in complexes of the Geometry package, primitives may not be coincident. Also similarly, adjacency is represented with boundary information, i.e. boundary and coboundary relationships. ISO 19107 states in the Topology package, when a primitive is completely surrounded by a primitive of at least two higher dimensions, no

boundary associations exist and the IsolatedIn association is used. Because the non-complex case is not available in the Topology package, as it is in the Geometry package, the IsolatedIn association does not correspond entirely with the InteriorTo association. Where the InteriorTo association has two interpretations, one for the non-complex case and one for the complex case, the IsolatedIn association has only the one interpretation corresponding to the complex case of the InteriorTo association.

For the InteriorTo and the IsolatedIn associations to correspond, in the two and three dimensional subprofiles, the InteriorTo association is constrained with the following constraint:

```
context GM_Primitive inv:
coincidentSubelement->forAll( sub | sub.dimension() <
                              self.dimension() - 1 )
```

This constraint requires two geometric primitives associated by the InteriorTo association to have a dimensional difference greater than 1. This constraint corresponds to the constraint expressed for the IsolatedIn association in ISO 19107:

```
context TP_Primitive inv:
isolated.dimension() < self.dimension() - 1
```

For the InteriorTo and the IsolatedIn associations to correspond, in the one dimensional subprofiles, the InteriorTo association can be simply constrained to 0 in both roles. Although the IsolatedIn association is not constrained it is effectively 0 for both roles as well. As shown above, ISO 19107 requires topological primitives associated by the IsolatedIn association to have a dimensional difference greater than 1, but the dimensional difference in the one dimensional profiles in no greater than one, rendering any constraint of the IsolatedIn association superfluous.

Annex K – Realization examples

(informative)

K.1 Introduction

K.1.1 Introduction

K.1.2 Purpose

This annex gives realization examples of selected profiles L1.2D.2d, L1.3D.3d, L3.2D.2d, L3.3D.3d, L4.2D.2d, L4.3D.3d, L6.2D.2d, and L6.3D.3d from Table 6.2.

The realization examples are based on a small area of model terrain with a few simple features shown in Figure K.1: a tower on a wooded hill, a river running through a valley, a road that crosses the river on a bridge and a building in a field with a short connection to the road. Figure K.2 shows a simple 2D map of these features.

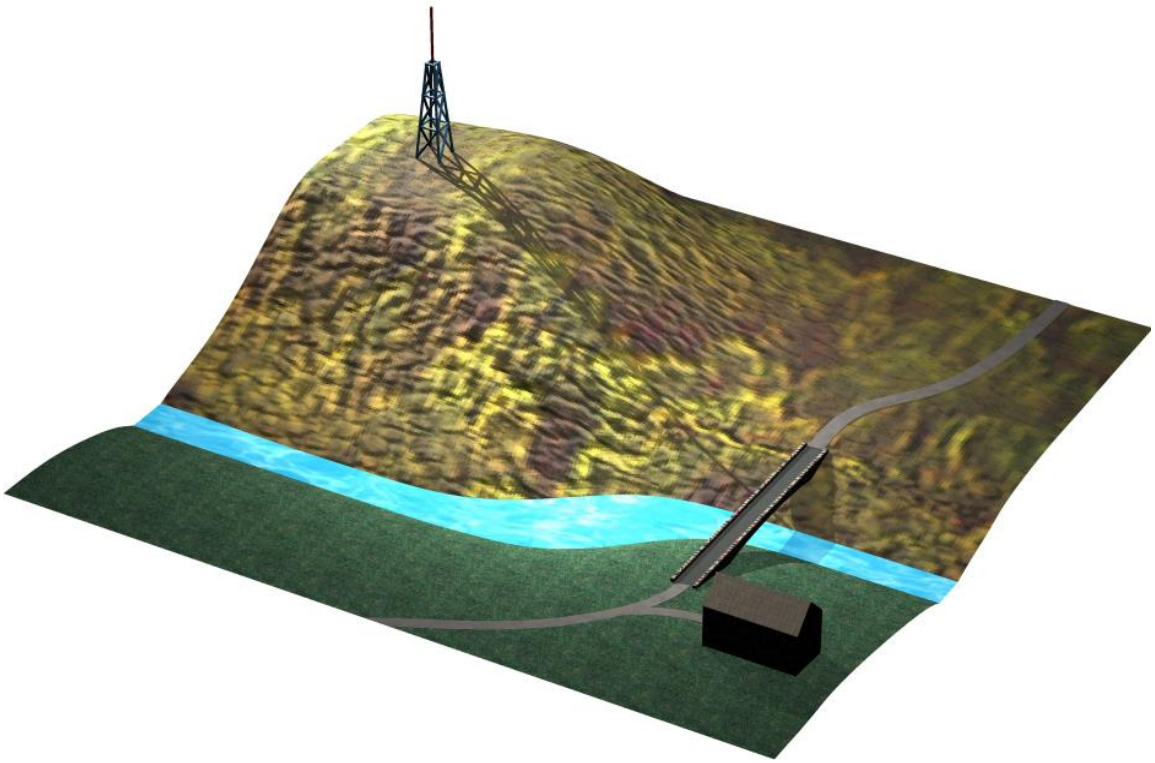


Figure K.1 – Example model terrain

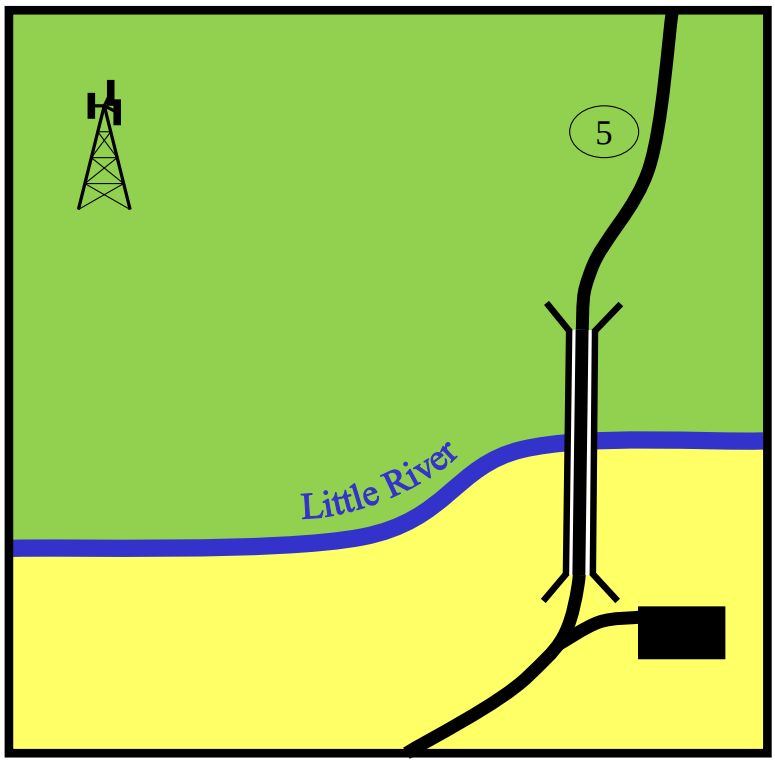


Figure K.2 – Map of example terrain

K.1.3 Application of profiles

The six levels of profiles define six levels of refinement of geospatial data. The first three levels define very crude data. Level 1 is completely free with no constraints on the geometric primitives. Level 2 requires that primitives do not self intersect. Level 3 requires that primitives do not self intersect and that primitives do not intersect each other, i.e. primitives are disjoint. This leads to boundary primitives being shared. Level 4 is transitional, adding the geometric primitives, which form the geometric complexes, on which the topological complexes of the following two levels are constructed. Levels 5 and 6 define refined data with the added value of full topological relationships, where level 6 fully partitions the 2D or 3D space level 5 is free to leave holes. The use of the first two levels is not recommended because the data can very quickly become inconsistent leading to gaps and slivers. On the other hand using levels 5 and 6 is recommended. David Galdi from the U. S. Bureau of the Census in his paper “Spatial Data Storage and Topology in the Redesigned MAF/TIGER System” clearly captures the reasons for producing geospatial data with topological relationships – note in particular the assertions regarding "persistent topology":

What is Topology?

Topology involves the mathematical study of spatial relationships. It describes the characteristics of a geometric figure that do not change under continuous transformation. In a graph, the number of line segments, intersection points, and polygons, and their relationship to each other, are constant as the plane in which they exist is stretched or distorted. In GIS applications, topology is the means to describe, manage, and retrieve these relationships explicitly without resorting to time-consuming spatial comparisons (Ramage and Woodsford 2002).

The principles of topology are utilized to implement a system that provides for:

- Rapid spatial data retrieval.
- Enhanced spatial data analysis.
- Spatial data editing and clean up.
- Improved data consistency.
- Management of shared geometry (Hoel 2003).
- Definition and enforcement of data integrity rules (Hoel 2003).

Some GIS applications use "persistent topology", that is, they structure the data according to topological principles so that the topological relationships are stored and available persistently in the database. In addition to the above benefits, this persistent topology approach also provides for:

- More efficient storage of spatial data.
- Reduction or elimination of redundancy of spatial data.
- Easier implementation of certain spatial business rules in the database.
- Improved management of hierarchical geographic relationships (Ramage and Woodsford 2002).
- A more straightforward way to address coordinate precision and tolerance issues that sometimes lead to gaps or slivers (Egenhofer et. al. 1989).

This document may be found in HTML format at http://www.census.gov/geo/mtep_obj2/topo_and_data_stor.html and as a PDF at http://www.census.gov/geo/mtep_obj2/topo_and_data_stor.pdf.

Each profile has applicability dependent on its profile level.

Level 1 and 2 profiles, 'free geometry' and 'no self intersection', have limited applicability. Since they should not be used in end products, if used at all, they should only be used as intermediate data structures in the production process.

Level 3 profiles, 'primitives disjoint', are suited for data sets used for graphical representations

Level 4 profiles, 'complex', are suited for graphical representation like Level 3. Moreover, Level 4 data sets are an intermediate step between Level 3 and Level 5, adding the geometric boundary primitives required by Level 5.

Level 5 and 6 profiles, 'full topology' and 'partitioned space', have the broadest applicability, being suited for both graphical representation and complex analysis. Analysis such as network analysis profits from boundary/coboundary relationships. Finding the edges that adjoin a given edge is accomplished by navigating the boundary relationships to the start and end nodes of the edge then in turn navigating the coboundary relationships of the nodes to determine their cobounding edges.

Each maximal primitive dimension has its own applicability. The 1D profiles can be used for network data: transportation networks, communication networks, etc. The 2D profiles can be used for most terrain representations. The 3D profiles can be used for high resolution volumetric data, as in urban settings, as well as geological and meteorological applications.

K.1.4 Structure of examples

Each example contains a number of graphics and diagrams. The first figure shows the geometric representations of the primitives used to represent the example terrain. The second figure is a diagram of how the geometric primitives relate to each other. This figure shows the primitives but not the boundary elements (i.e. Rings and Shells). Lines connecting primitives show boundary/coboundary relationships. Arrowheads show navigability. The third figure is a UML diagram showing the topological structures of the realization. The remaining figures of each example show how each primitive and each boundary element realizes the ISO 19107 types.

K.1.5 Realization

The reader will note that the realization classes look different than the ISO 19107 types. It is important to understand what a realization is. In "The Unified Modelling Language Reference Manual, Second Edition" by James Rumbaugh, Ivar Jacobson and Grady Booch it explains that:

> The meaning of realization is that the client element must support all the behaviour of the supplier element but need not match its structure or implementation.

It goes on to say that:

> But any attributes, associations, methods, or state machines of the supplier that specify implementation are irrelevant to the client.

Thus each realization is not a verbatim implementation of the elements of the profile but a schema that implements the functionality of the profile.

K.1.6 Class names

Class names used for the realizations are examples only and may be changed for an actual implementation.

K.2 Profile Realization Examples

K.2.1 L1 – Free Geometry

K.2.1.1 Introduction

The profiles in this group specify collections of geometric primitives. The primitives are not guaranteed to be disjoint and may self-intersect. The primitives do not form complexes. The primitive are independent and do not relate to each other, except the Surface class in 2D is associated with its bounding Curves, and the Solid class in 3D is associated with its bounding Surfaces. This group of profiles can be useful if all that is required is a visual image or plot of a map and no spatial analysis is to be performed.

K.2.1.2 2D Free Geometry

Figure K.3 shows the geometric primitives that represent the features in Figure K.1, as implemented using the 2D Free Geometry profile. Figure K.4 is an instance diagram showing the topological relationships between the example primitives. The only topological relationships represented in this profile are surfaces referencing their bounding curves since, in this profile, this is how surfaces are defined. Figure K.5 is a UML model showing the topological relationships among the primitive types. Here again, only surfaces are topologically related to curves, through rings. Figure K.6, Figure K.7, Figure K.8, and Figure K.9 show the realizations of the Point, Curve, Ring, and Surface geometric objects respectively.

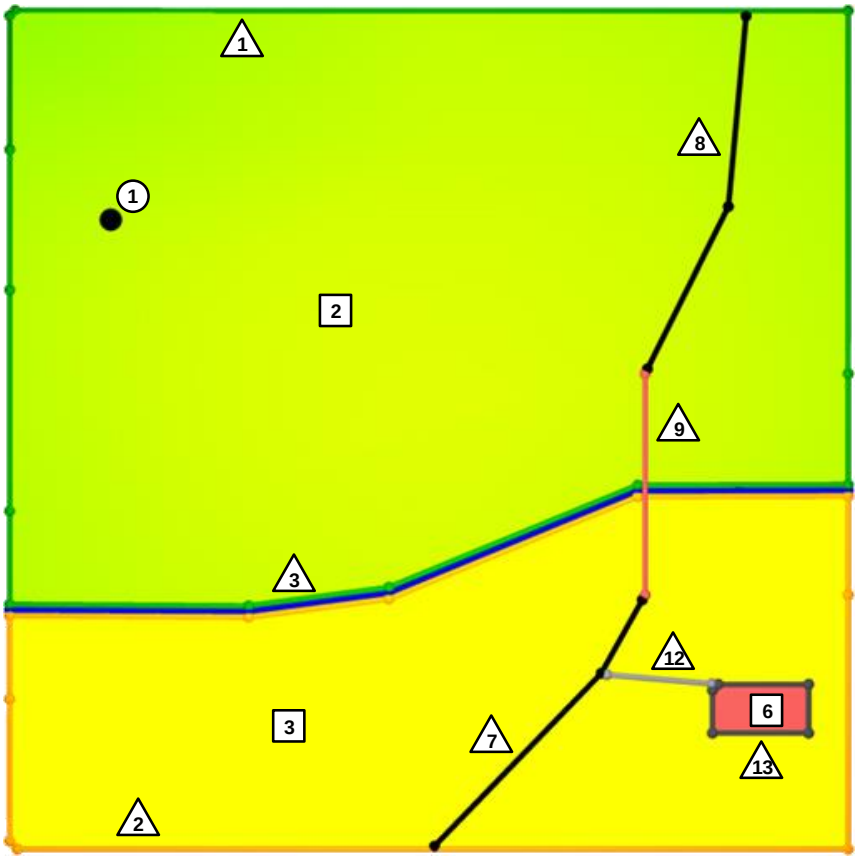


Figure K.3 – View of 2D primitives

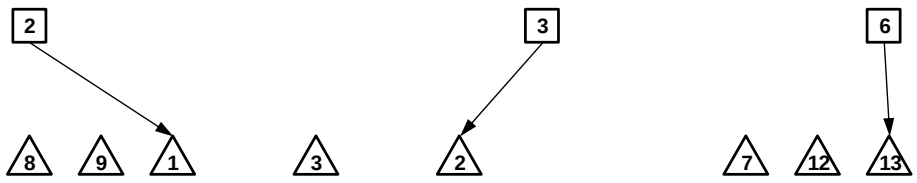


Figure K.4 – Instance diagram

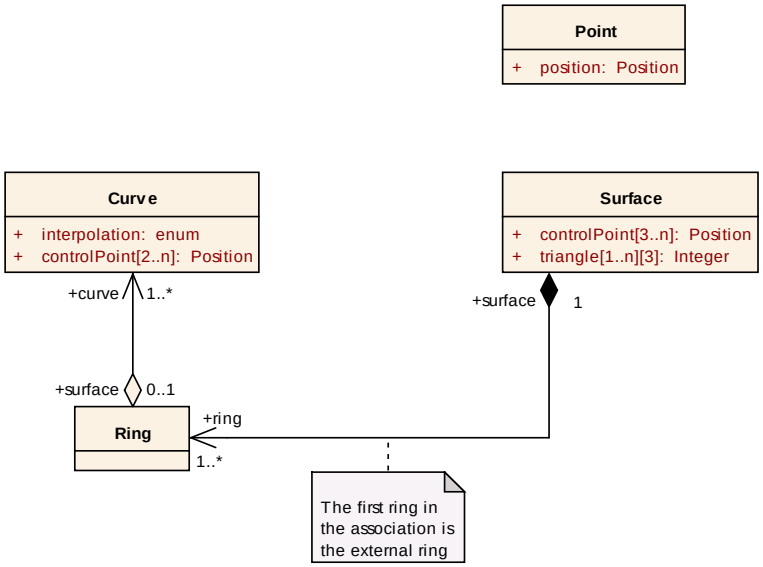


Figure K.5 – Topological relationships

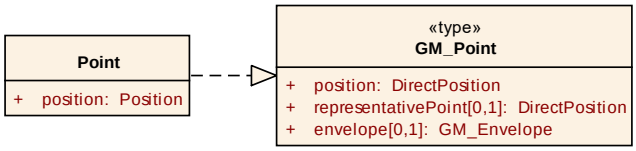


Figure K.6 – Point realization

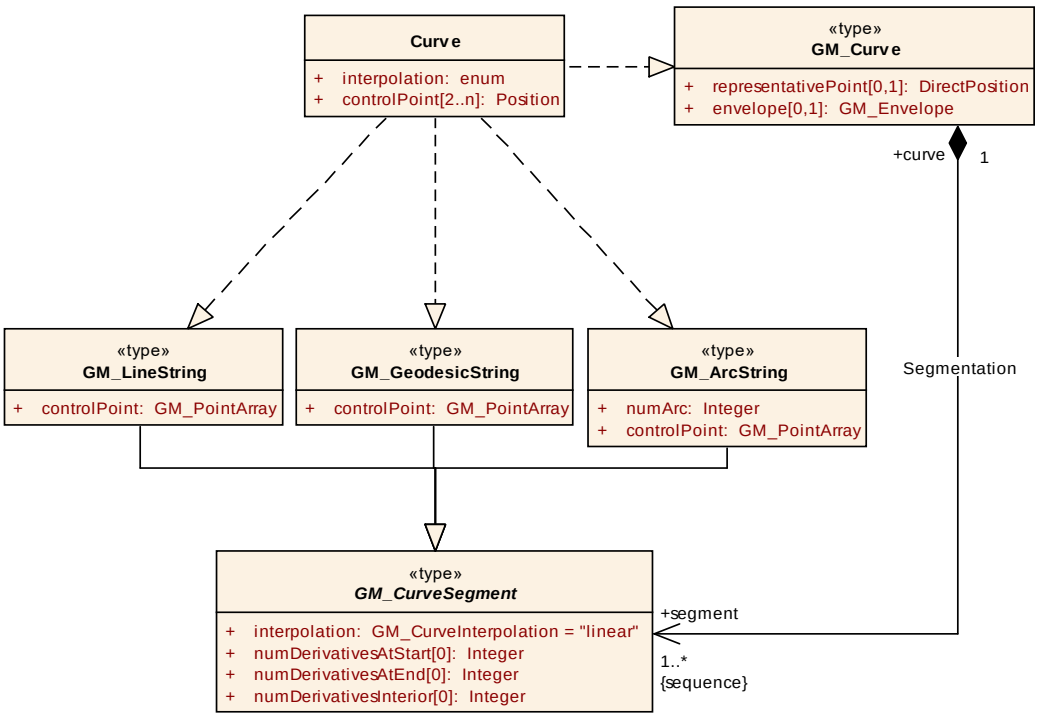


Figure K.7 – Curve realization

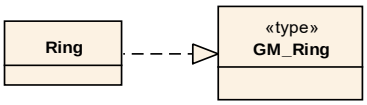


Figure K.8 – Ring realization

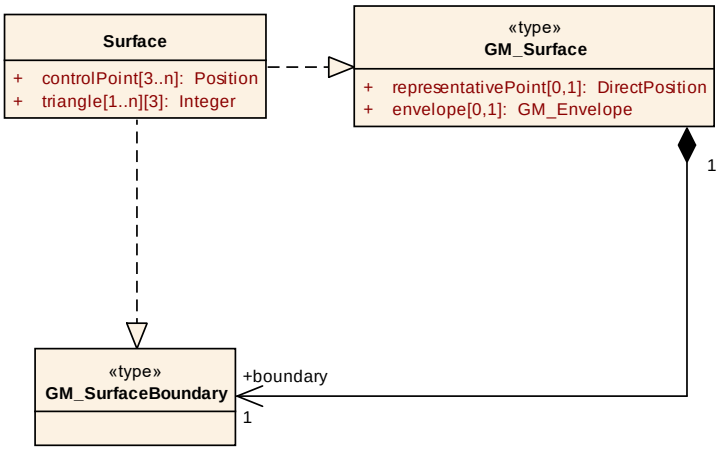


Figure K.9 – Surface realization

K.2.1.3 3D Free Geometry

Figure K.10 shows the geometric primitives that represent the features in Figure K.1, as implemented using the 3D Free Geometry profile. The two area features are each represented by a TIN surface (each labelled with a numbered square). Note that the curves representing the roads do not necessarily conform to the terrain surface. Figure K.11 is an instance diagram showing the topological relationships between the example primitives. The only topological relationships represented in this profile are solids referencing their bounding surfaces since, in this profile, this is how solids are defined. Solid “2”, representing the building, is bounded by surface “3”. Figure K.12 is a UML model showing the topological relationships among the primitive types. Here again, only solids are topologically related to surfaces, through shells. Figure K.13 (identical to Figure K.6) shows the realization of Point geometric objects. Figure K.14 (identical to Figure K.7) shows the realization of Curve geometric objects. Figure K.15, which adds a TIN and removes the surface boundary from Figure K.9, shows the realization of Surface geometric objects. Figure K.16 and Figure K.17 show the realizations of the Shell, and Solid geometric objects respectively.

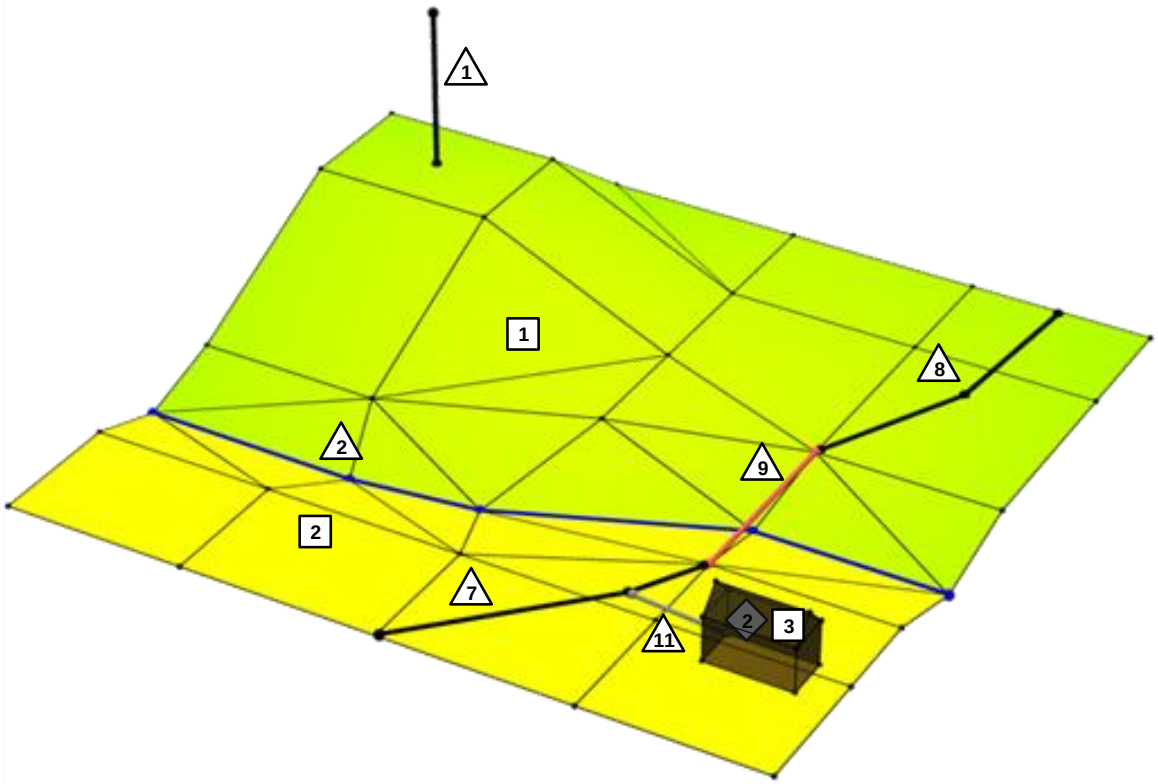


Figure K.10 – View of 3D primitives

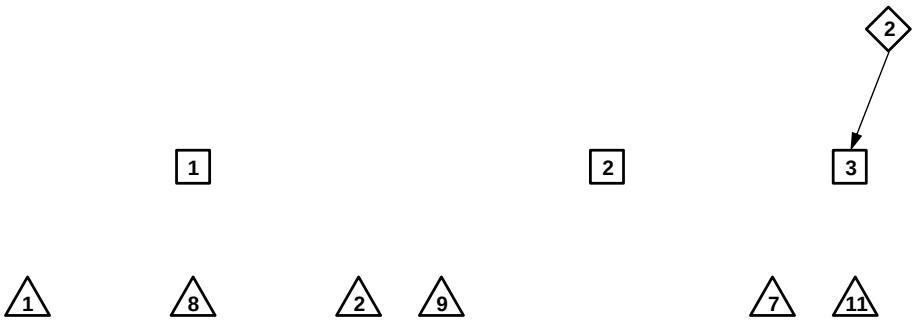


Figure K.11 – Instance diagram

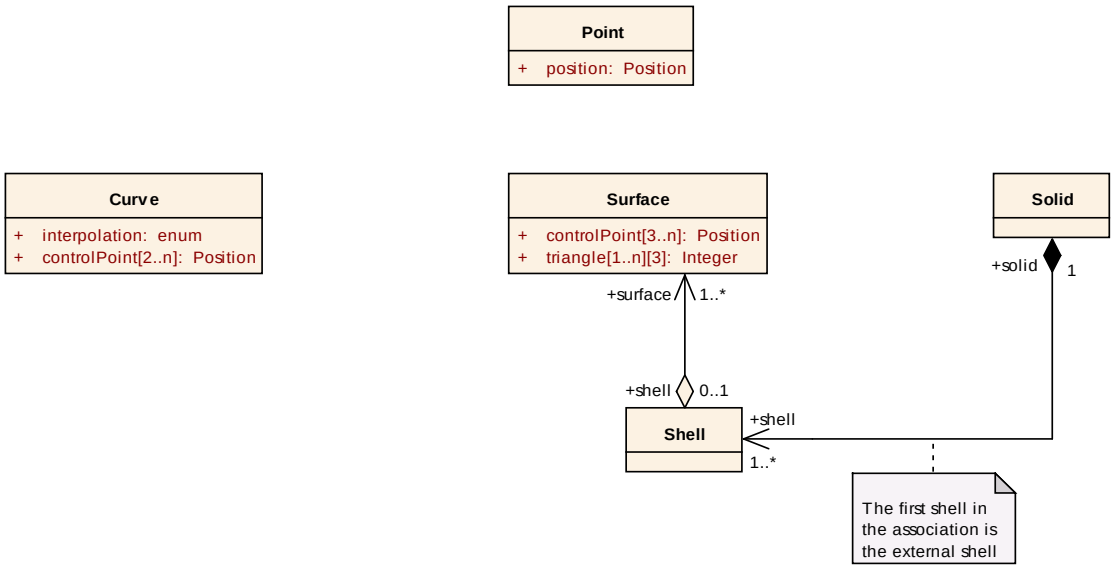


Figure K.12 – Topological relationships

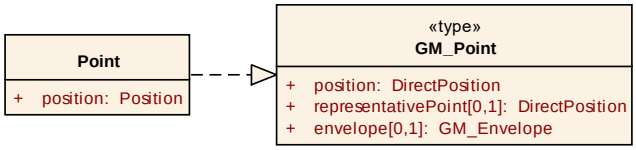


Figure K.13 – Point realization

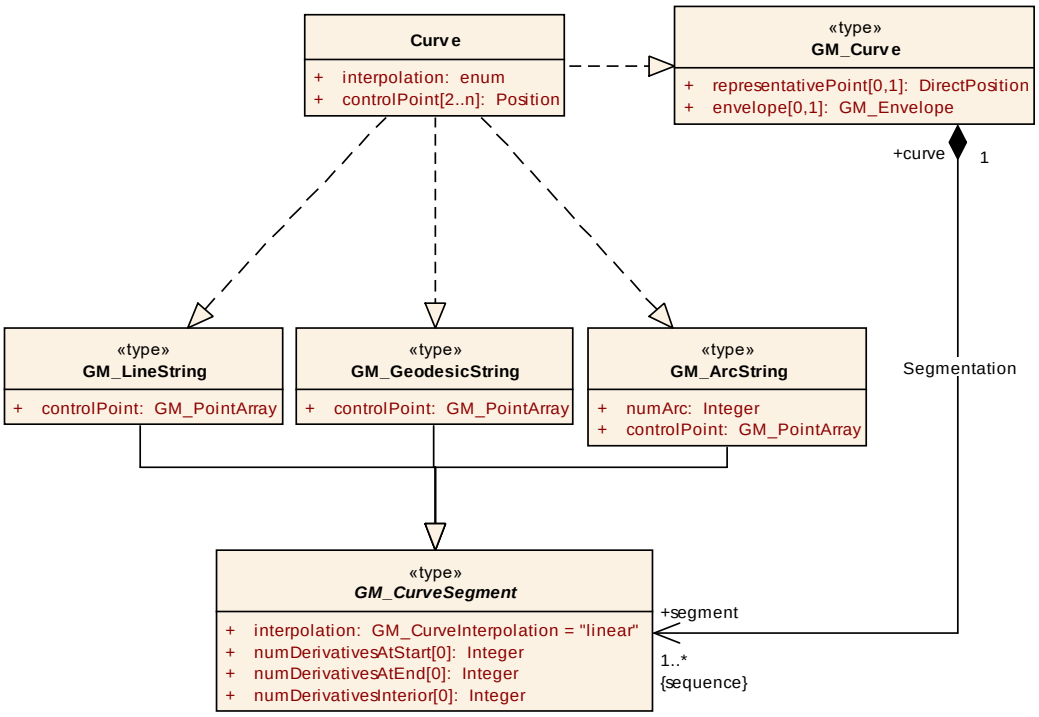


Figure K.14 – Curve realization

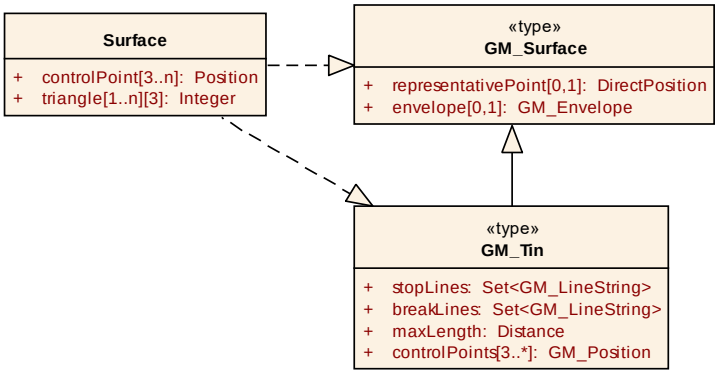


Figure K.15 – Surface realization

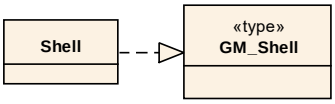


Figure K.16 – Shell realization

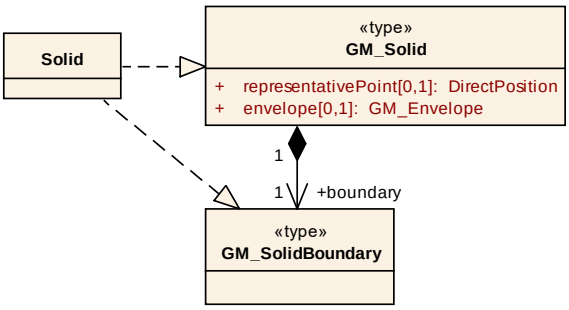


Figure K.17 – Solid realization

K.2.2 L3 – Primitives Disjoint

K.2.2.1 Introduction

The profiles in this group specify collections of geometric primitives. In contrast to the profiles described in the previous section, the primitives in these profiles **are** guaranteed to be disjoint and may **not** self-intersect, but the primitives still do not form complexes. As in the previous group of profiles, primitive are independent and do not relate to each other, except the Surface class in 2D is associated with its bounding Curves, and the Solid class in 3D is associated with its bounding Surfaces. This group of profiles can be useful if all that is required is a visual image or plot of a map and no spatial analysis is to be performed and is the first step in cleaning up the spaghetti data of the profiles in the previous section.

K.2.2.2 2D Shared

Figure K.18 shows the geometric primitives that represent the features in Figure K.1, as implemented using the 2D Shared profile. In this profile surface boundary curves are shared between surface rings. These shared curves have associated directionality. Curves that intersected in the 2D Free Geometry profile are split at the intersection in this profile. Figure K.19 is an instance diagram showing the topological relationships between the example primitives. The only topological relationships represented in this profile are surfaces referencing their bounding curves since, in this profile, this is how surfaces are defined. Figure K.20 is a UML model showing the topological relationships among the primitive types. This diagram is similar to Figure K.5 adding directionality to the Ring-Curve association. Here again, only surfaces are topologically related to curves, through rings. Figure K.21 (identical to Figure K.6) shows the realization of Point geometric objects. Figure K.22 shows the realization of Curve geometric objects, adding

GM_OrientableCurve to Figure K.7. Figure K.23 (identical to Figure K.8) shows the realization of Ring geometric objects. Figure K.24 (identical to Figure K.9) shows the realization of Surface geometric objects.

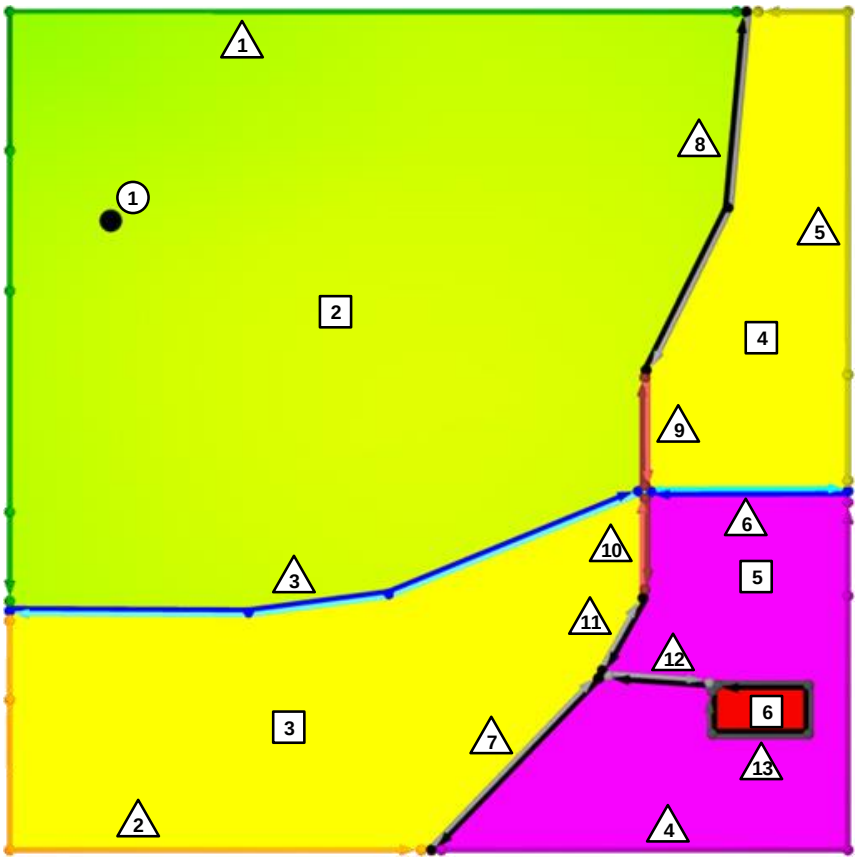


Figure K.18 – View of 2D primitives

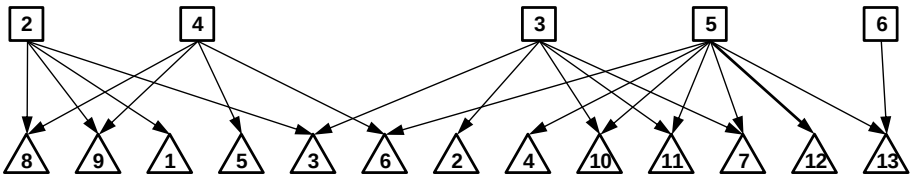


Figure K.19 – Instance diagram

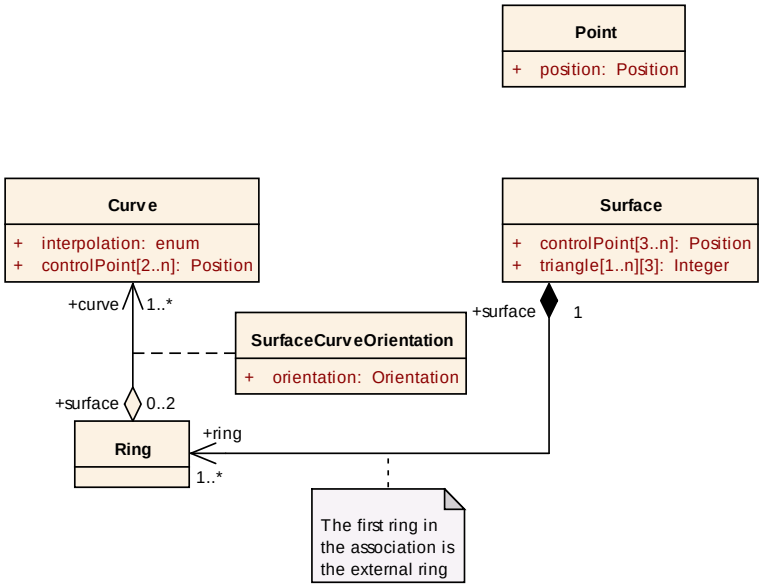


Figure K.20 – Topological relationships

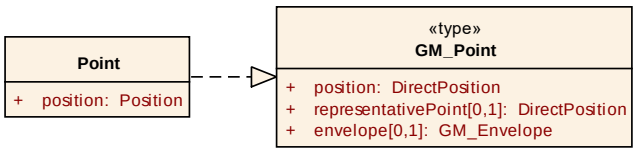


Figure K.21 – Point realization

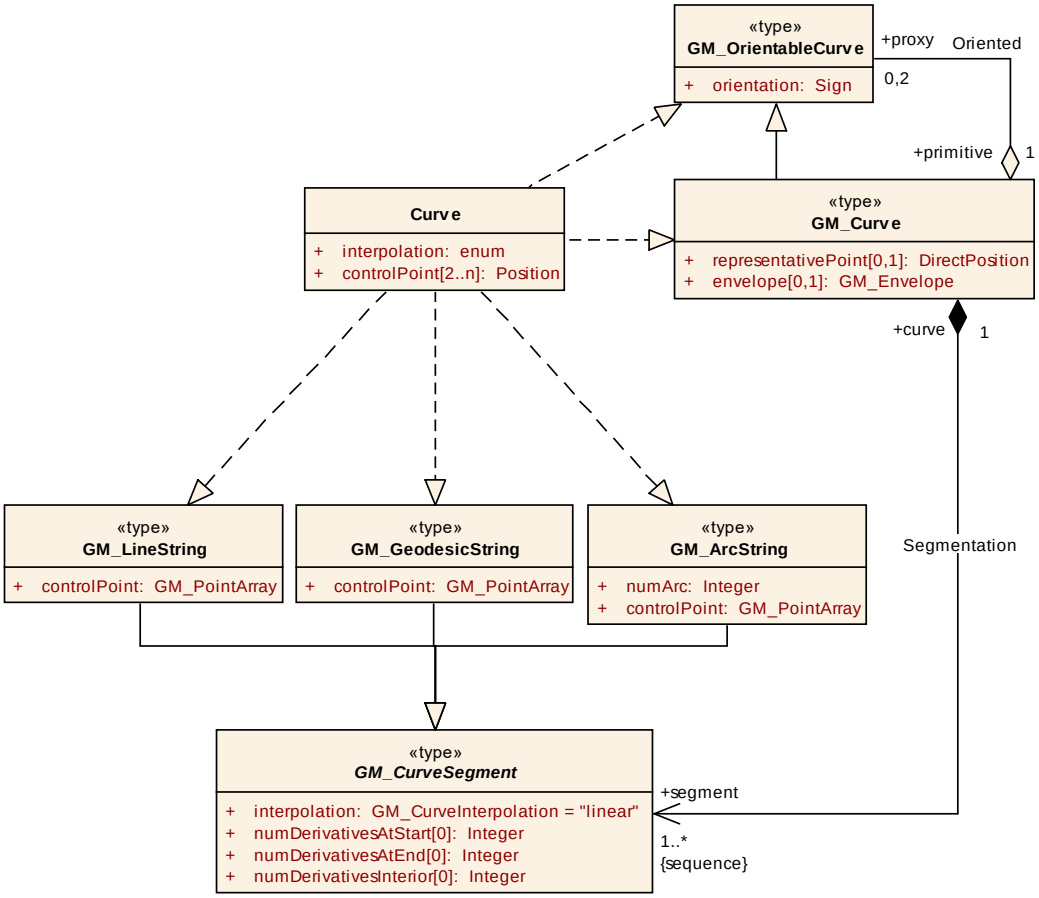


Figure K.22 – Curve realization

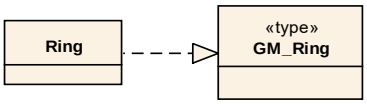


Figure K.23 – Ring realization

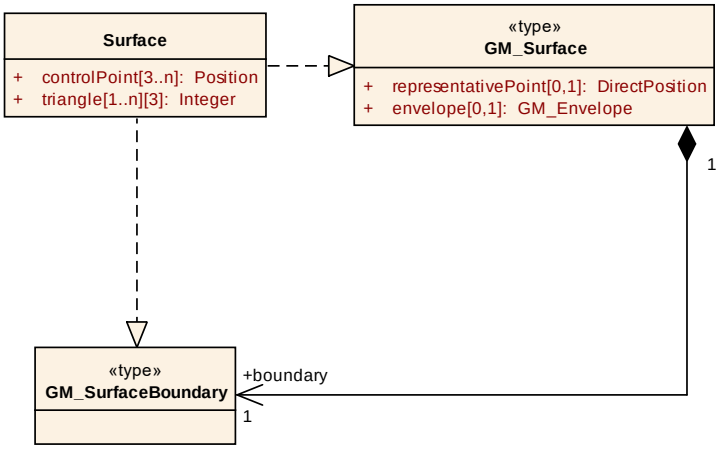


Figure K.24 – Surface realization

K.2.2.3 3D Shared

Figure K.25 shows the geometric primitives that represent the features in Figure K.1, as implemented using the 3D Shared profile. The two area features are each represented by a TIN surface (each labelled with a numbered square). The curves representing the roads conform to the terrain surface. In this profile solid boundary surfaces are shared between solid shells. These shared surfaces have associated directionality. Figure K.26 is an instance diagram showing the topological relationships between the example primitives. The only topological relationships represented in this profile are solids referencing their bounding surfaces since, in this profile, this is how solids are defined. Solid "2", representing the building, is bounded by surfaces "3" and "4"; which conforms to the terrain. Figure K.27 is a UML model showing the topological relationships among the primitive types. This diagram is similar to Figure K.12 adding directionality to the Shell-Surface association. Here again, only solids are topologically related to surfaces, through shells. Figure K.28 (identical to Figure K.13) shows the realization of Point geometric objects. Figure K.29 (identical to Figure K.22) shows the realization of Curve geometric objects. Figure K.30 shows the realization of Surface geometric objects, adding GM_OrientableSurface to Figure K.15. Figure K.31 (identical to Figure K.16) shows the realization of Shell geometric objects. Figure K.32 (identical to Figure K.17) shows the realization of Solid geometric objects.

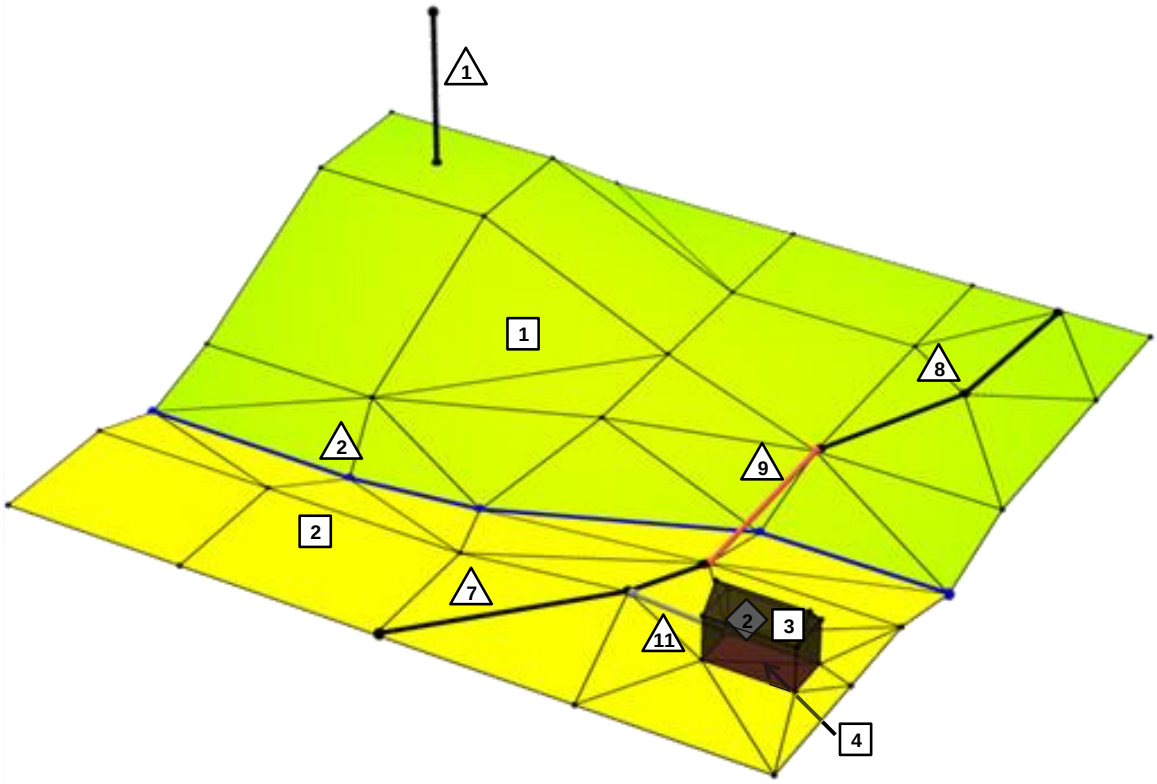


Figure K.25 – View of 3D primitives

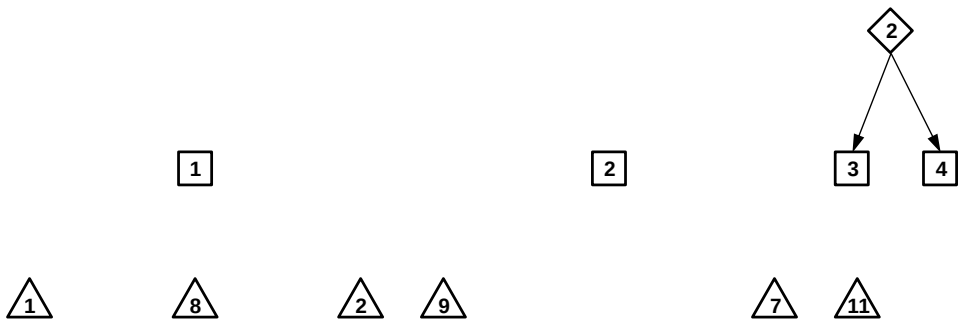


Figure K.26 – Instance diagram

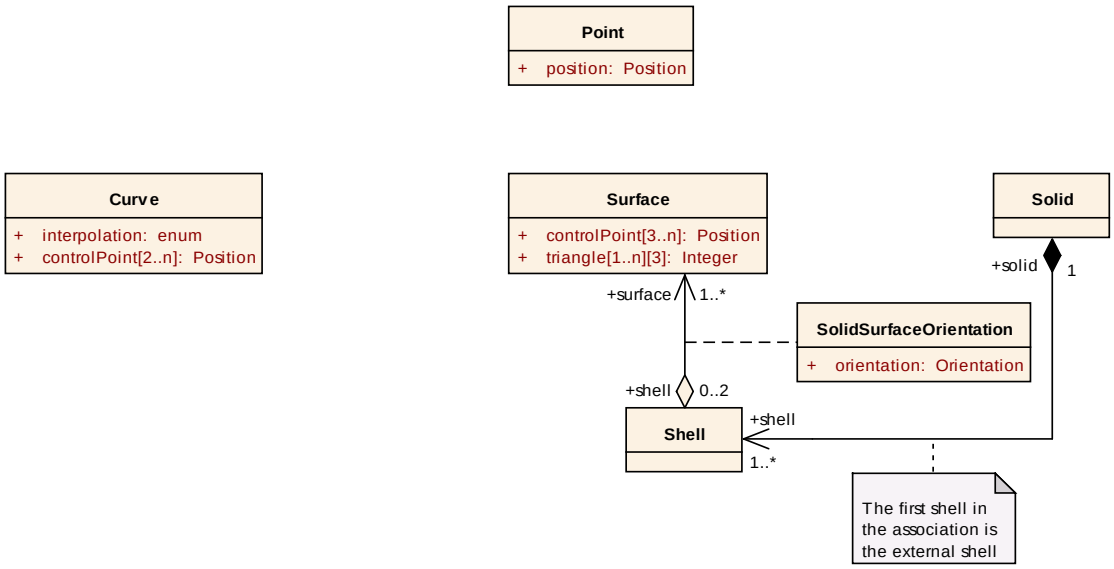


Figure K.27 – Topological relationships

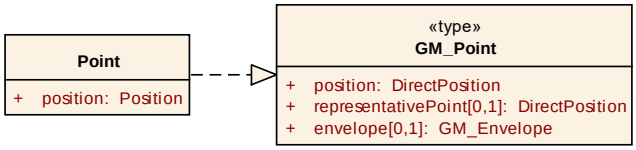


Figure K.28 – Point realization

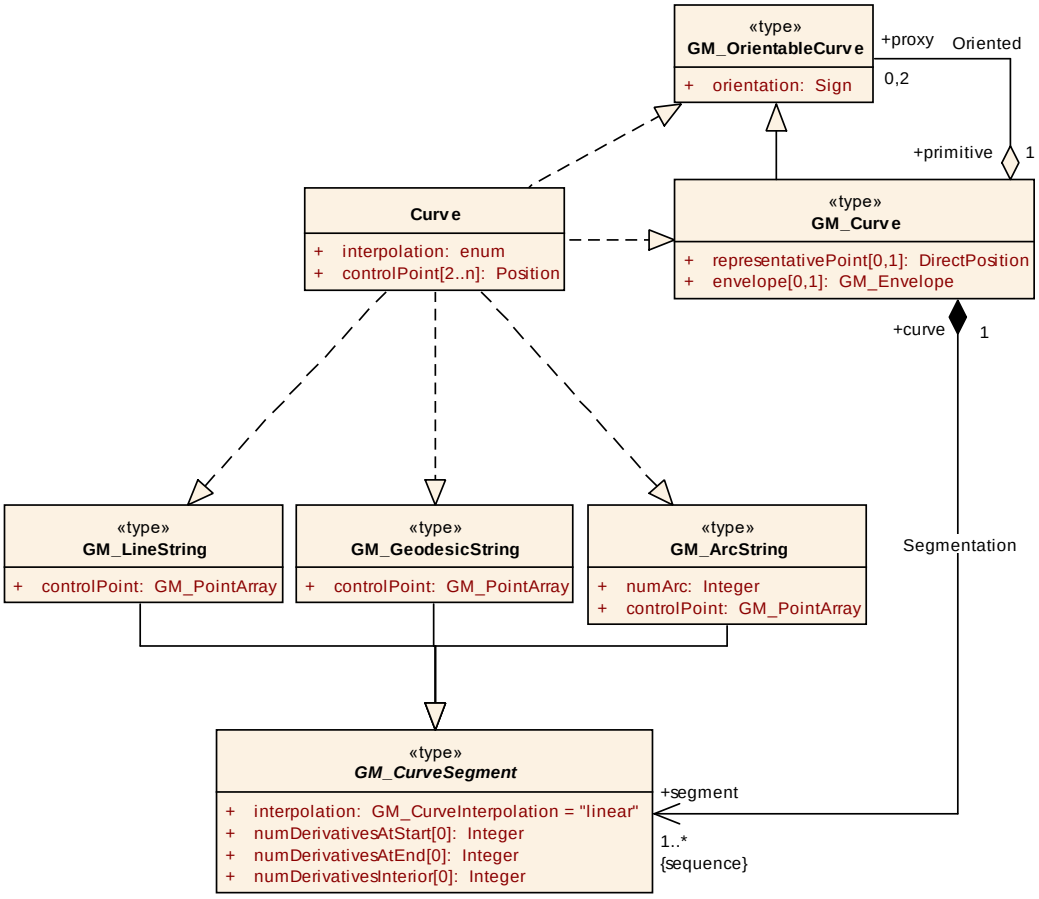


Figure K.29 – Curve realization

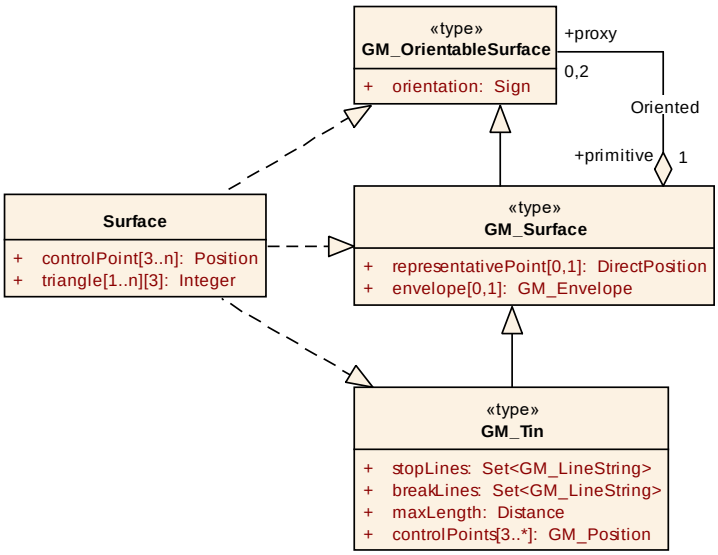


Figure K.30 – Surface realization

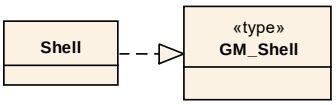


Figure K.31 – Shell realization

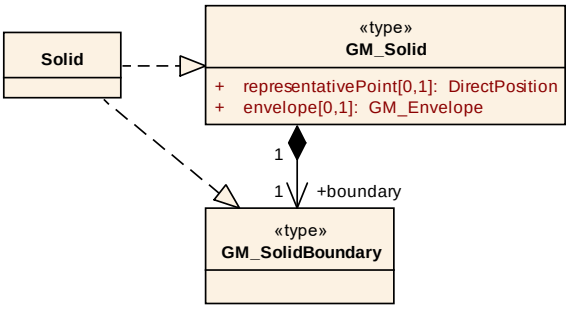


Figure K.32 – Solid realization

K.2.3 L4 – Complex

K.2.3.1 Introduction

The profiles in this group specify complexes of geometric primitives. That is to say, the interiors of the primitives are disjoint, and all boundary primitives exist. As complexes, all boundary objects exist. As in the previous groups of profiles, primitives are independent and do not relate to each other, except the Surface class in 2D is associated with its bounding Curves, and the Solid class in 3D is associated with its bounding Surfaces. These profiles further refine the profiles of the previous group. Having all boundary primitives is a prerequisite to establishing topological relationships. This group of profiles can be useful in an application, which builds topology on the fly.

K.2.3.2 2D Complex

Figure K.33 shows the geometric primitives that represent the features in Figure C.1, as implemented using the 2D Complex profile. In this profile surface boundary curves are shared between surface rings. These shared curves have associated directionality. All boundary points are added to the 2D Shared profile example to conform to the requirement of this profile to have all boundary primitives represented. Figure K.34 is an instance diagram showing the topological relationships between the example primitives. The only topological relationships represented in this profile are surfaces referencing their bounding curves since, in this profile, this is how surfaces are defined. Figure K.35 (identical to Figure K.20) is a UML model showing the topological relationships among the primitive types. Here again, only surfaces are topologically related to curves, through rings. Figure K.36 (identical to Figure K.21), Figure K.37 (identical to Figure K.22), Figure K.38 (identical to Figure K.23), and Figure K.39 (identical to Figure K.24) show the realizations of the Point, Curve, Ring, and Surface geometric objects respectively.

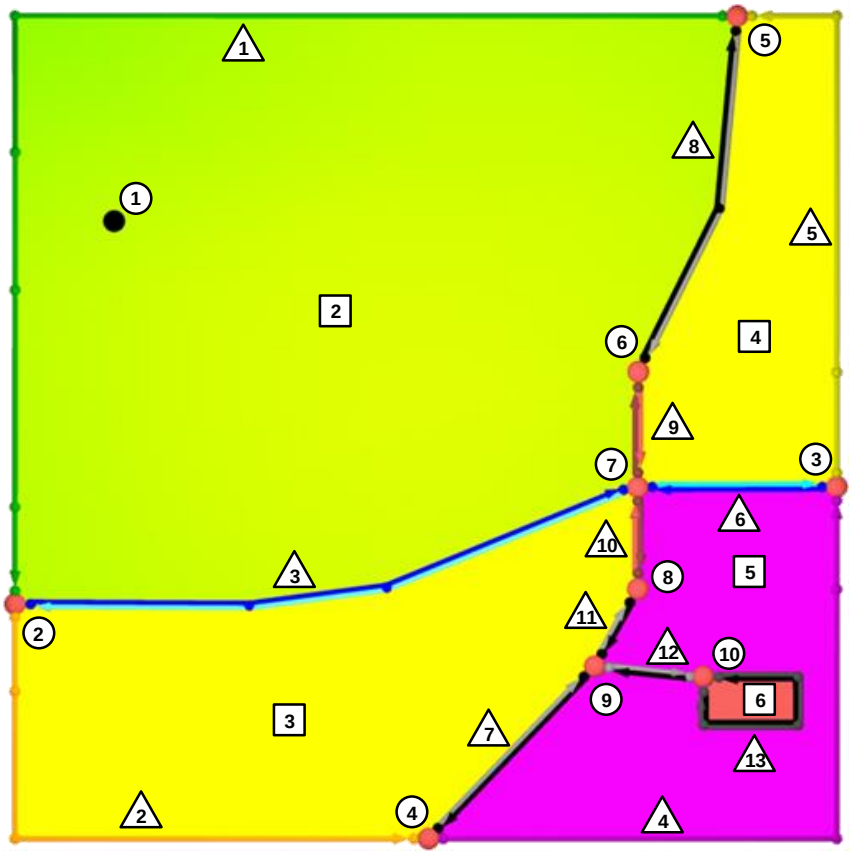


Figure K.33 – View of 2D primitives

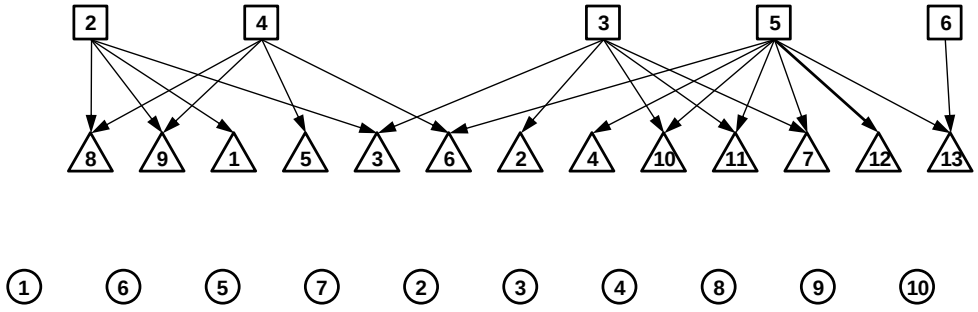


Figure K.34 – Instance diagram

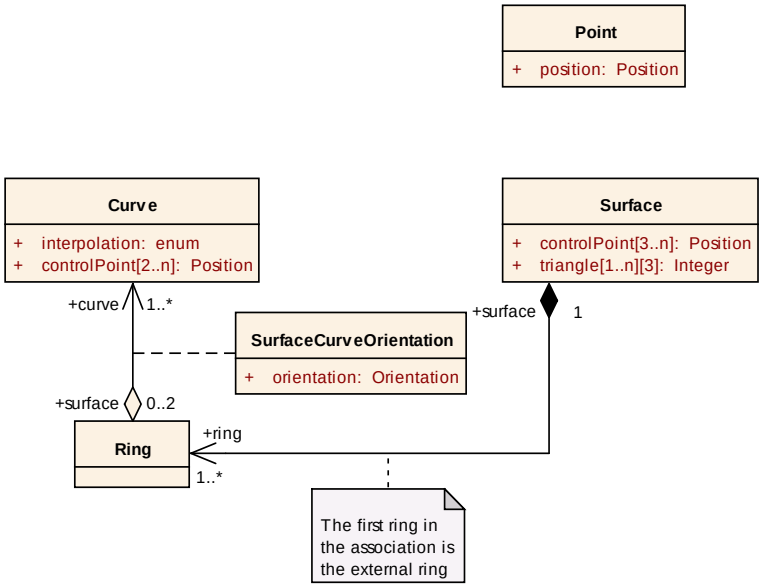


Figure K.35 – Topological relationships

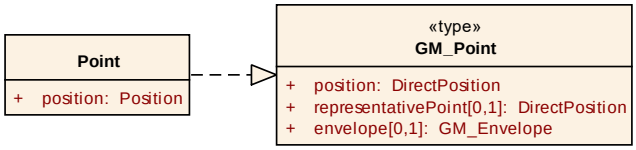


Figure K.36 – Point realization

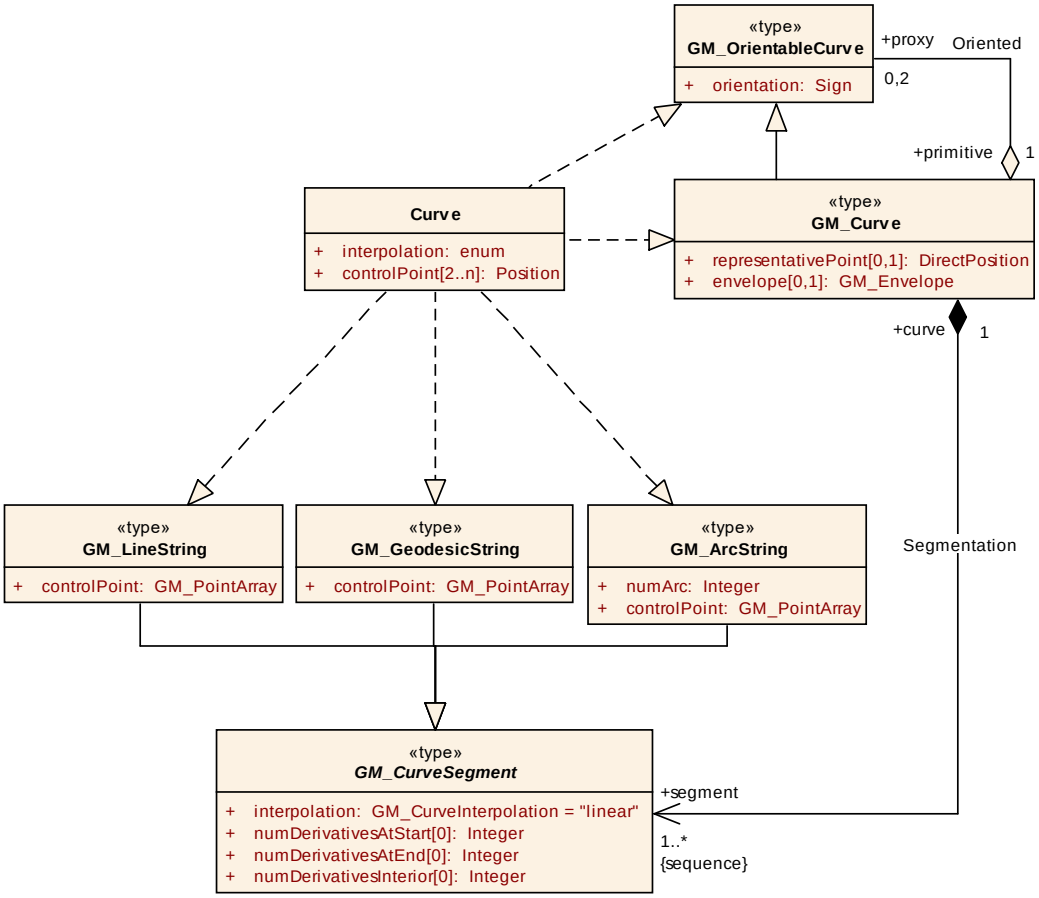


Figure K.37 – Curve realization

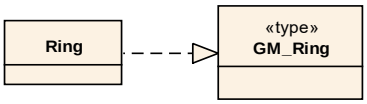


Figure K.38 – Ring realization

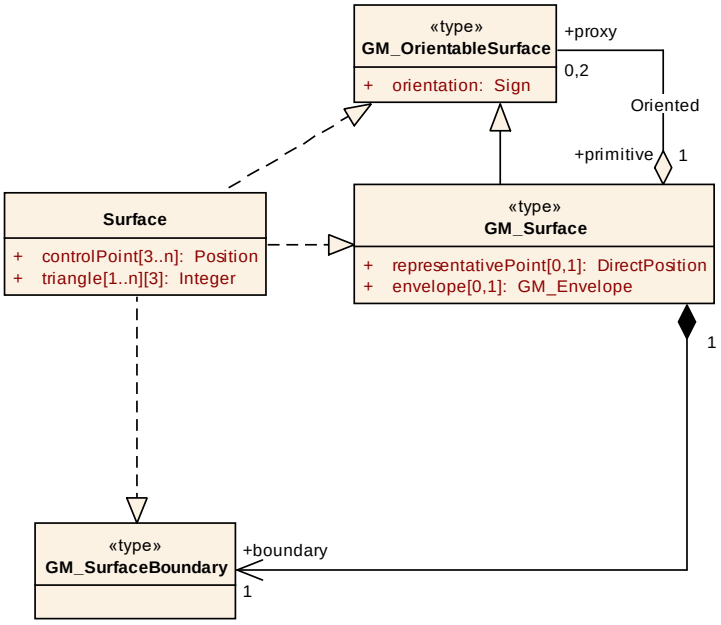


Figure K.39 – Surface realization

K.2.3.3 3D Complex

Figure K.40 shows the geometric primitives that represent the features in Figure K.1, as implemented using the 3D Complex profile. In this profile solid boundary surfaces are shared between solid shells. These shared surfaces have associated directionality. All boundary points, and curves with the appropriate directionality are added to the 3D Shared profile example to conform to the requirement of this profile to have all boundary primitives represented. Figure K.41 is an instance diagram showing the topological relationships between the example primitives. The only topological relationships represented in this profile are solids referencing their bounding surfaces since, in this profile, this is how solids are defined. Solid “2”, representing the building, is bounded by surfaces “3” and “4”. Surface “3” represents the top and sides of the building, while surface “4” represents the bottom of the building, which is integrated with the terrain surface. Curve “12” represents the footprint of the building, and forms the boundary between surfaces “2”, “3”, and “4”. Figure K.42 (identical to Figure K.27) is a UML model showing the topological relationships among the primitive types. As in the previous 3D profiles, only solids are topologically related to surfaces, through shells. Figure K.43 (identical to Figure C.28), Figure K.44 (identical to Figure C.29), Figure K.45 (identical to Figure C.30), Figure K.46 (identical to Figure C.31), and Figure K.47 (identical to Figure C.32) show the realizations of Point, Curve, Surface, Shell, and Solid geometric objects respectively.

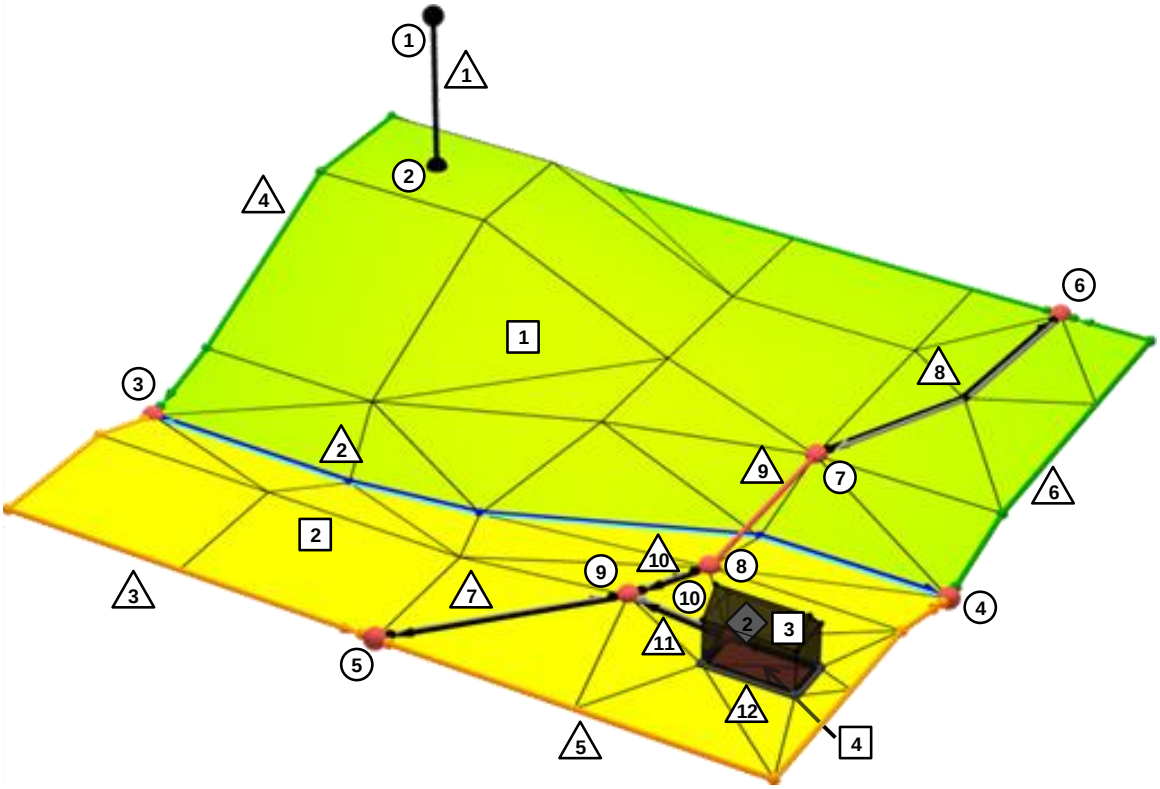


Figure K.40 – View of 3D primitives

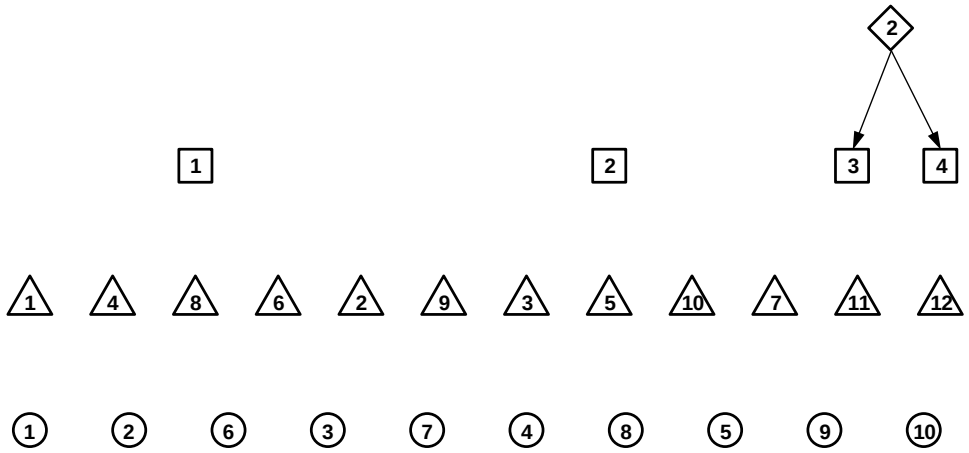


Figure K.41 – Instance diagram

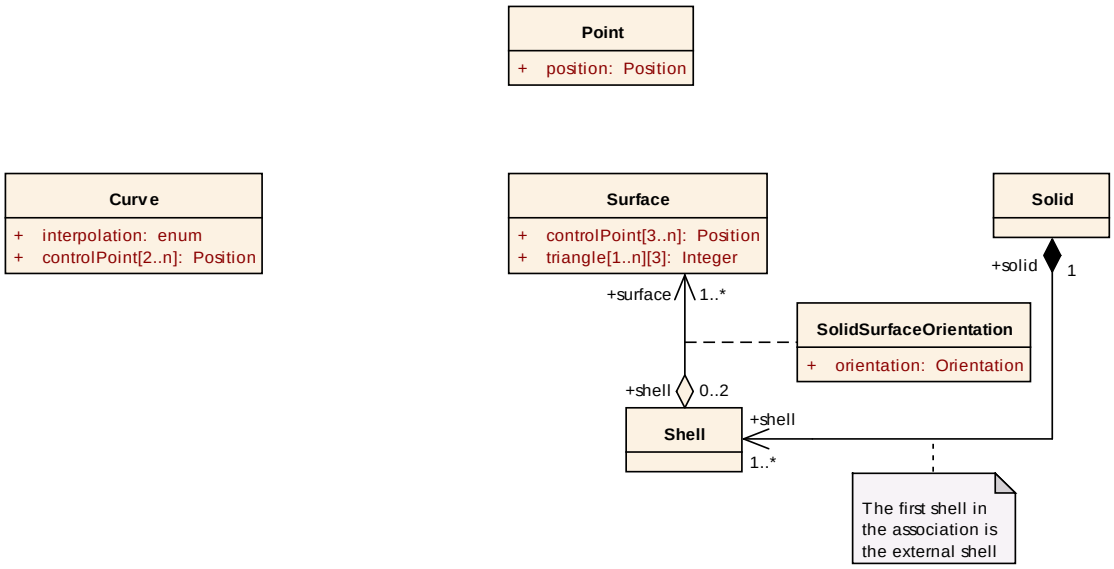


Figure K.42 – Topological relationships

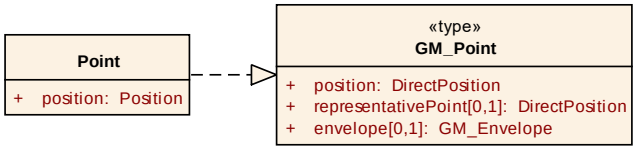


Figure K.43 – Point realization

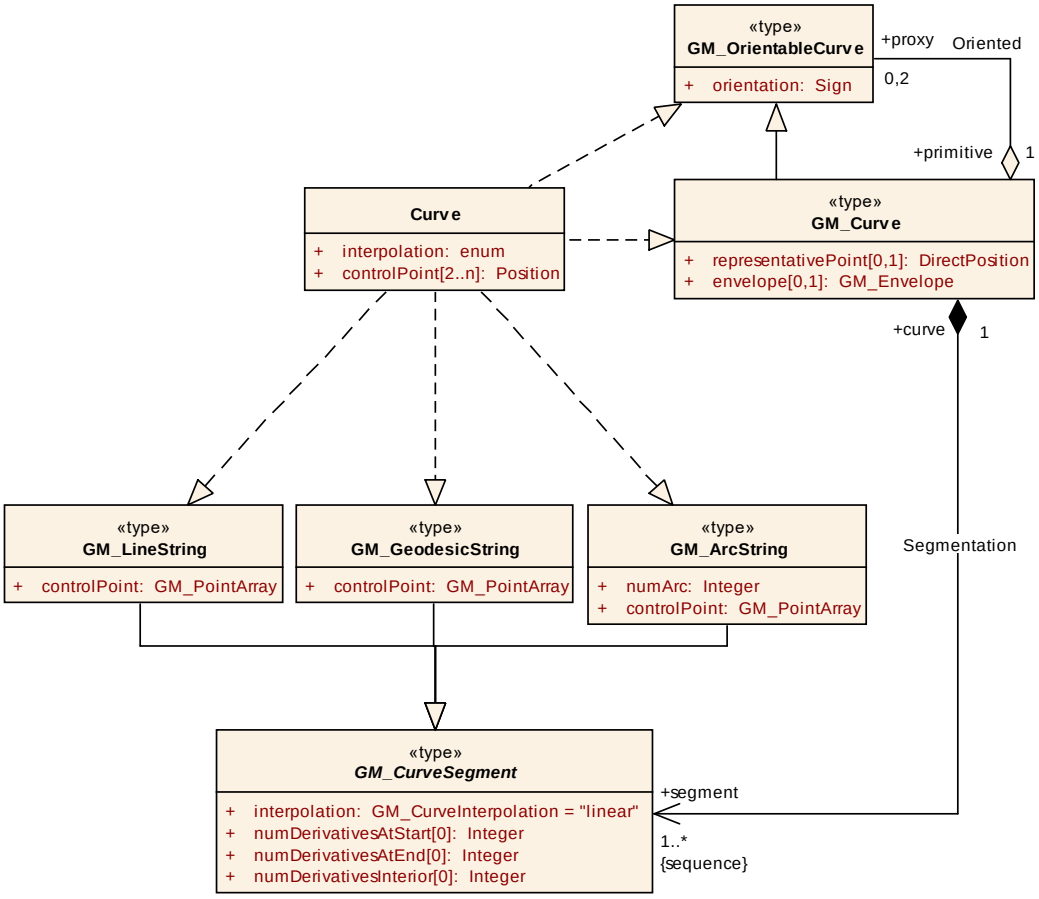


Figure K.44 – Curve realization

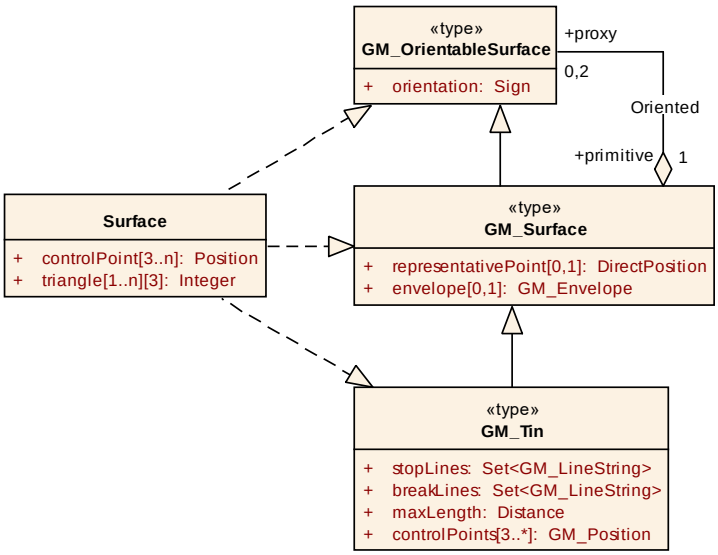


Figure K.45 – Surface realization

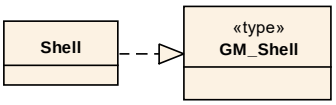


Figure K.46 – Shell realization

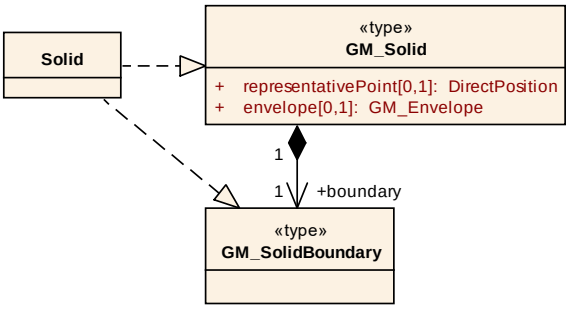


Figure K.47 – Solid realization

K.2.4 L6 – Partitioned Space

K.2.4.1 Introduction

The profiles in this group specify complexes of geometric primitives that partition 2D or 3D space with full topological relationships. As in the previous group the interiors of the primitives are disjoint and all boundary primitives exist. These profiles build on the profiles of the previous group linking primitives in boundary, coboundary, and containment relationships. This group of profiles can be useful in applications, which perform spatial analysis.

K.2.4.2 2D Tessellation

Figure K.48 shows the geometric/topological primitives that represent the features in Figure K.2, as implemented using the 2D Tessellation profile. In this profile primitives are in a complex; all primitives are disjoint and all boundary objects exist. This profile adds all boundary, coboundary, and containment associations to the 2D Complex profile as well as bidirectionality on all primitives. Figure K.49 is an instance diagram showing the paired boundary and coboundary relationships among the primitive instances, between the surfaces and their bounding curves, and between the curves and their bounding points. Surface “1” is the universe surface. The dashed line represents the containment relationship between the surface and the isolated point that it contains. Figure K.50 is a UML model showing the topological relationships among the primitive types. Figure K.51, Figure K.52, Figure K.53, and Figure K.54 show the realizations of the Point, Curve, Ring, and Surface geometric/topological objects respectively.

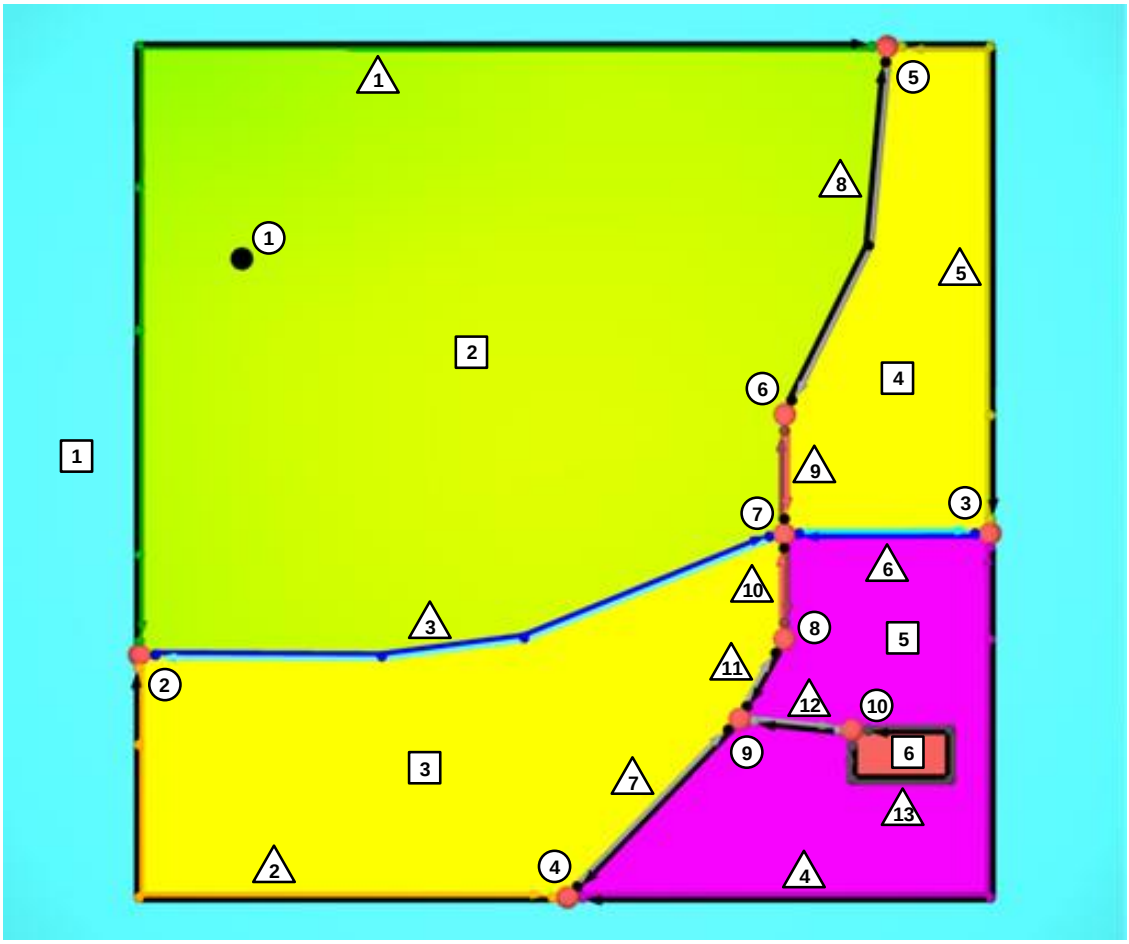


Figure K.48 – View of 2D primitives

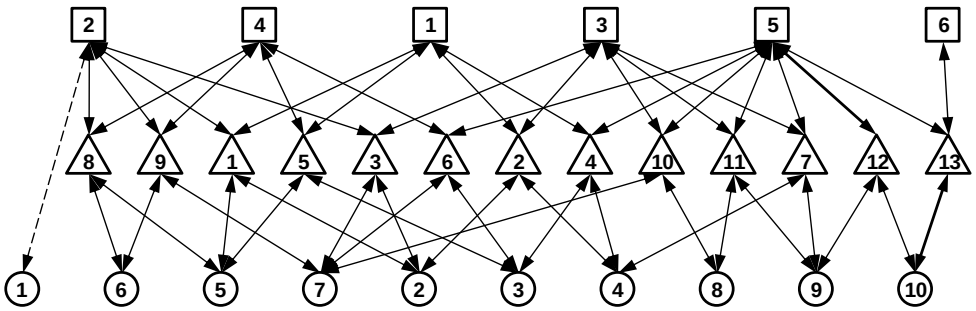


Figure K.49 – Instance diagram

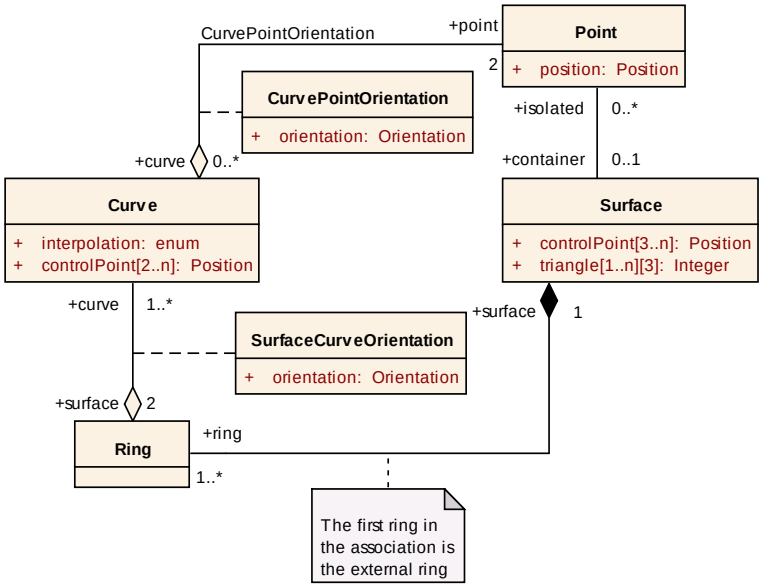


Figure K.50 – Topological relationships

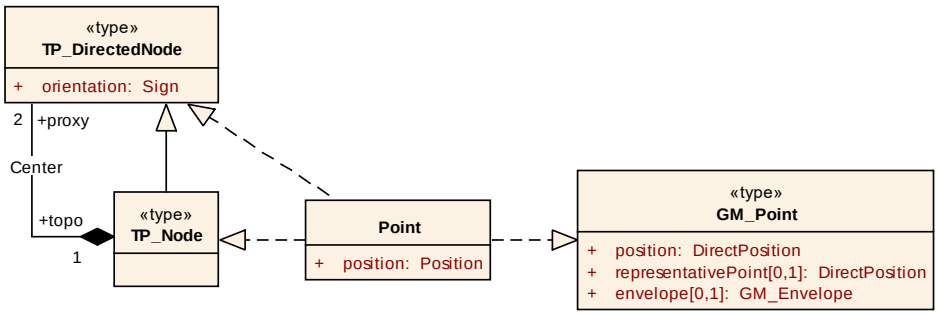


Figure K.51 – Point realization

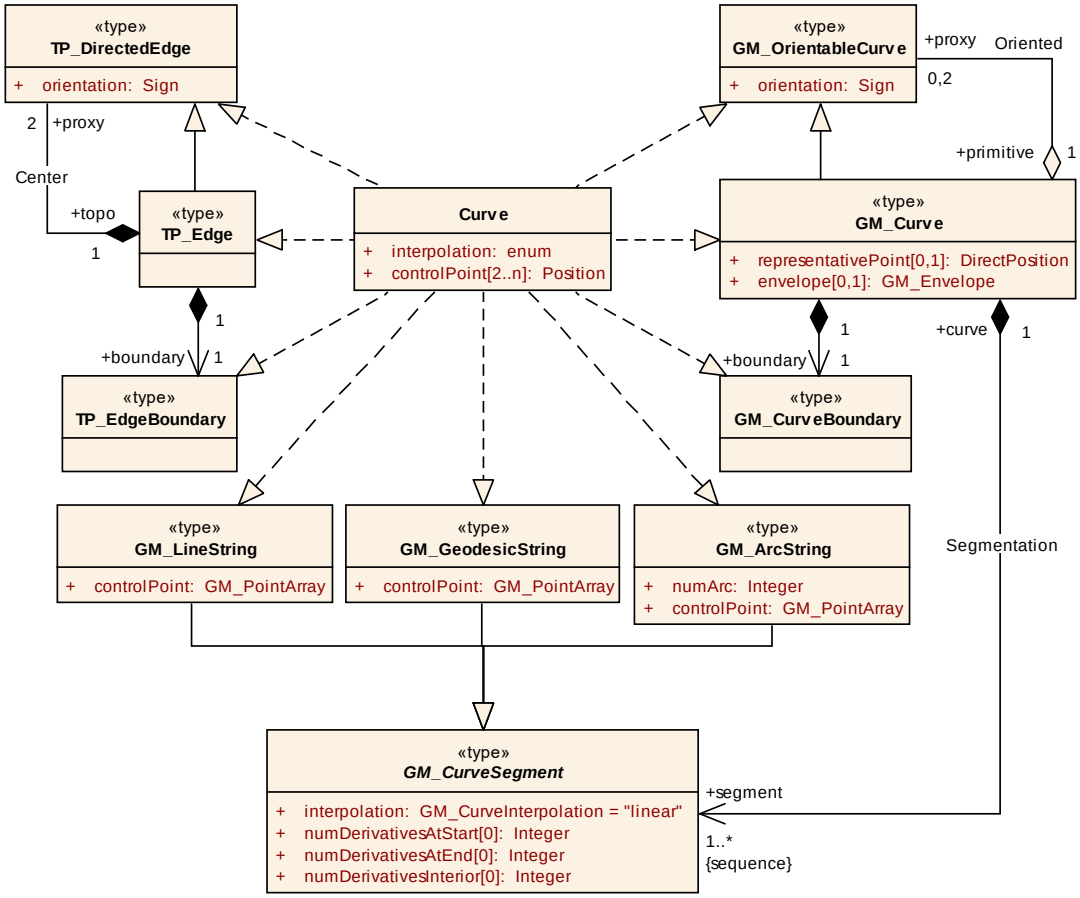


Figure K.52 – Curve realization

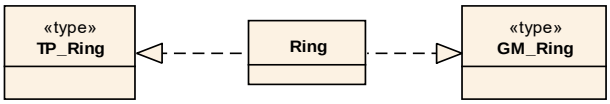


Figure K.53 – Ring realization

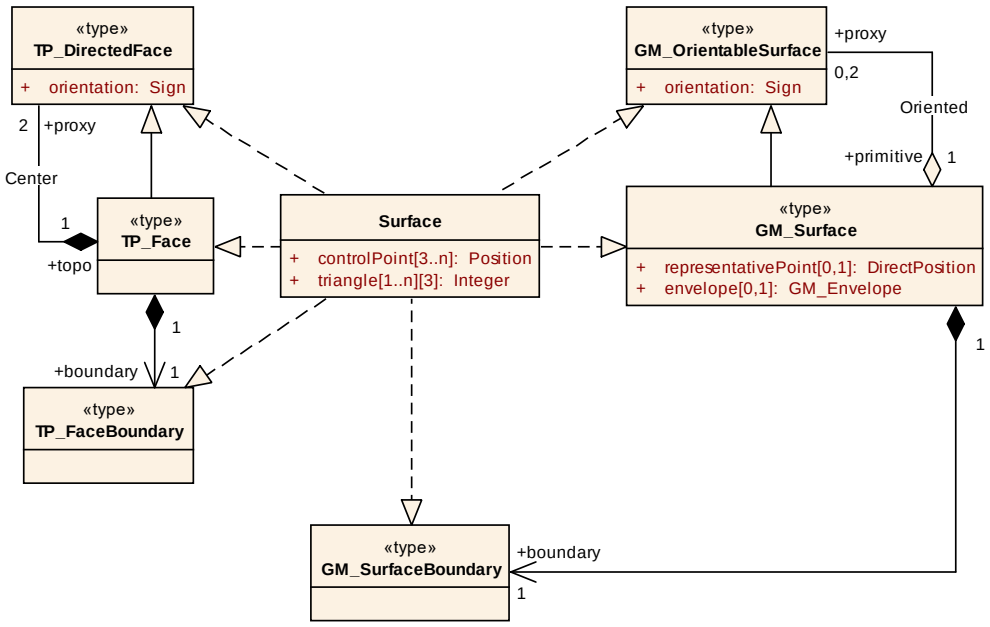


Figure K.54 – Surface realization

K.2.4.3 3D Tessellation

Figure K.55 shows the geometric/topological primitives that represent the features in Figure K.1, as implemented using the 3D Tessellation profile. In this profile primitives are in a complex; all primitives are disjoint and all boundary objects exist. This profile adds all boundary, coboundary, and containment associations to the 3D Complex profile as well as bidirectionality on all primitives. Figure K.56 is an instance diagram showing the topological relationships between the example primitives. Solid “2”, representing the building, is bounded by surfaces “3” and “4”. Surface “3” represents the top and sides of the building, while surface “4” represents the bottom of the building, which is integrated with the terrain surface. Curve “12” represents the footprint of the building, and forms the boundary between surfaces “2”, “3”, and “4”. Solid “1” is the universe solid. The dashed lines represent the containment relationships between surfaces and isolated points, between solids and isolated points , and between solids and isolated lines. Figure K.57 is a UML model showing the topological relationships among the primitive types. Figure K.58, Figure K.59, Figure K.60, Figure K.61, Figure K.62, and Figure K.63 show the realizations of the Point, Curve, Ring, Surface, Shell, and Solid geometric objects respectively.

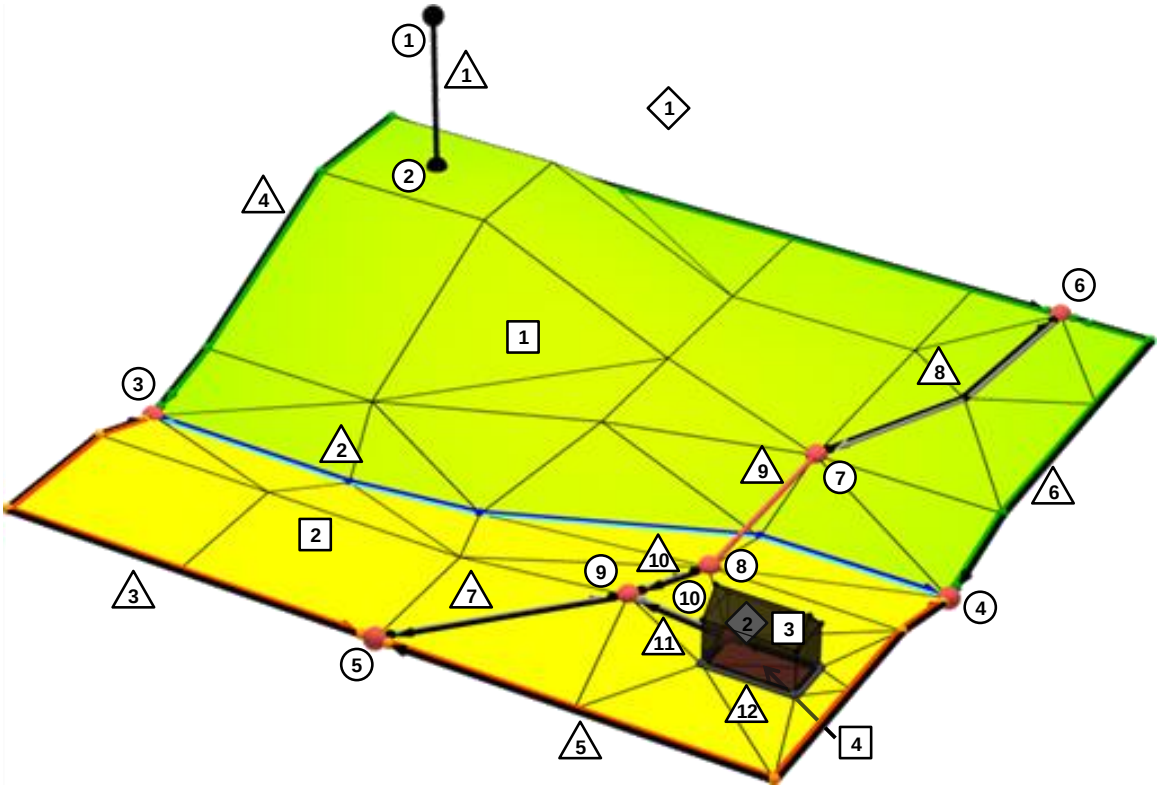


Figure K.55 – View of 3D primitives

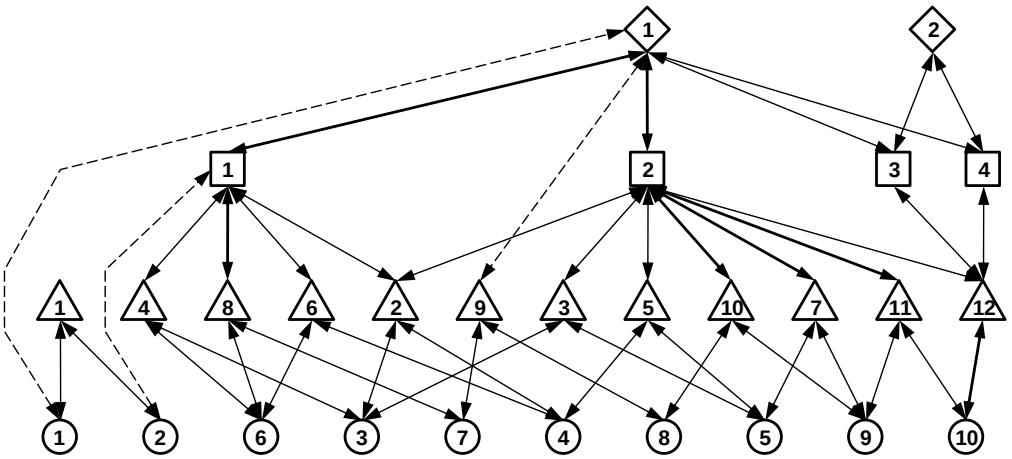


Figure K.56 – Instance diagram

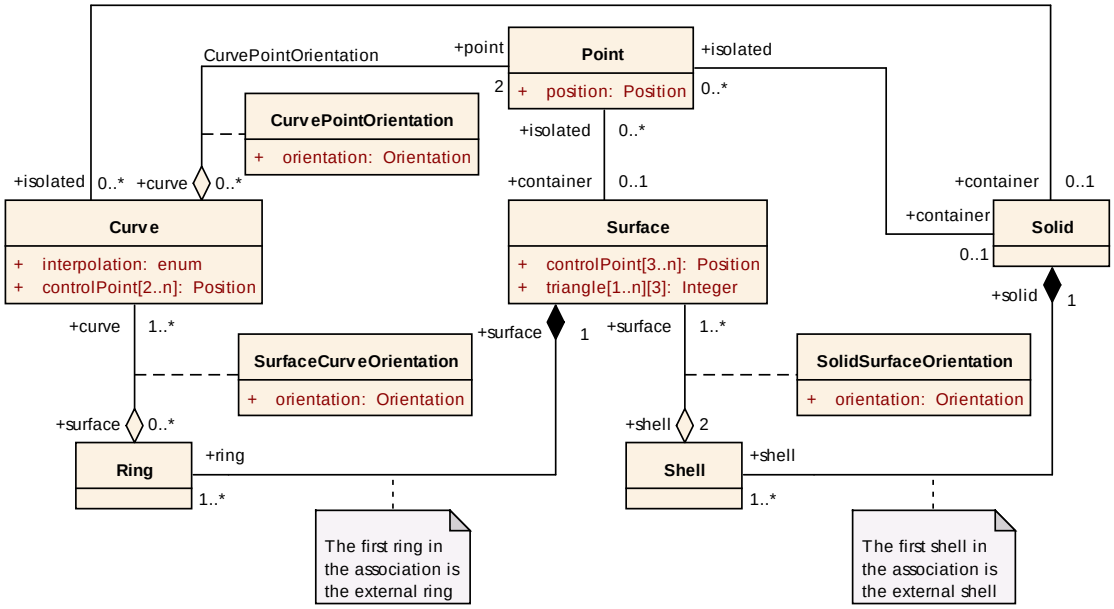


Figure K.57 – Topological relationships

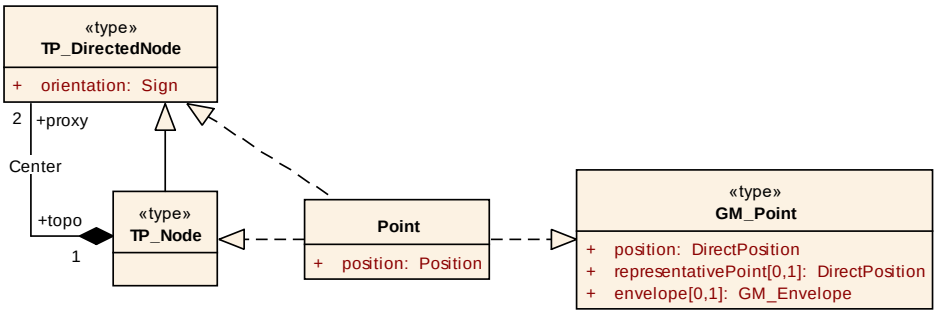


Figure K.58 – Point realization

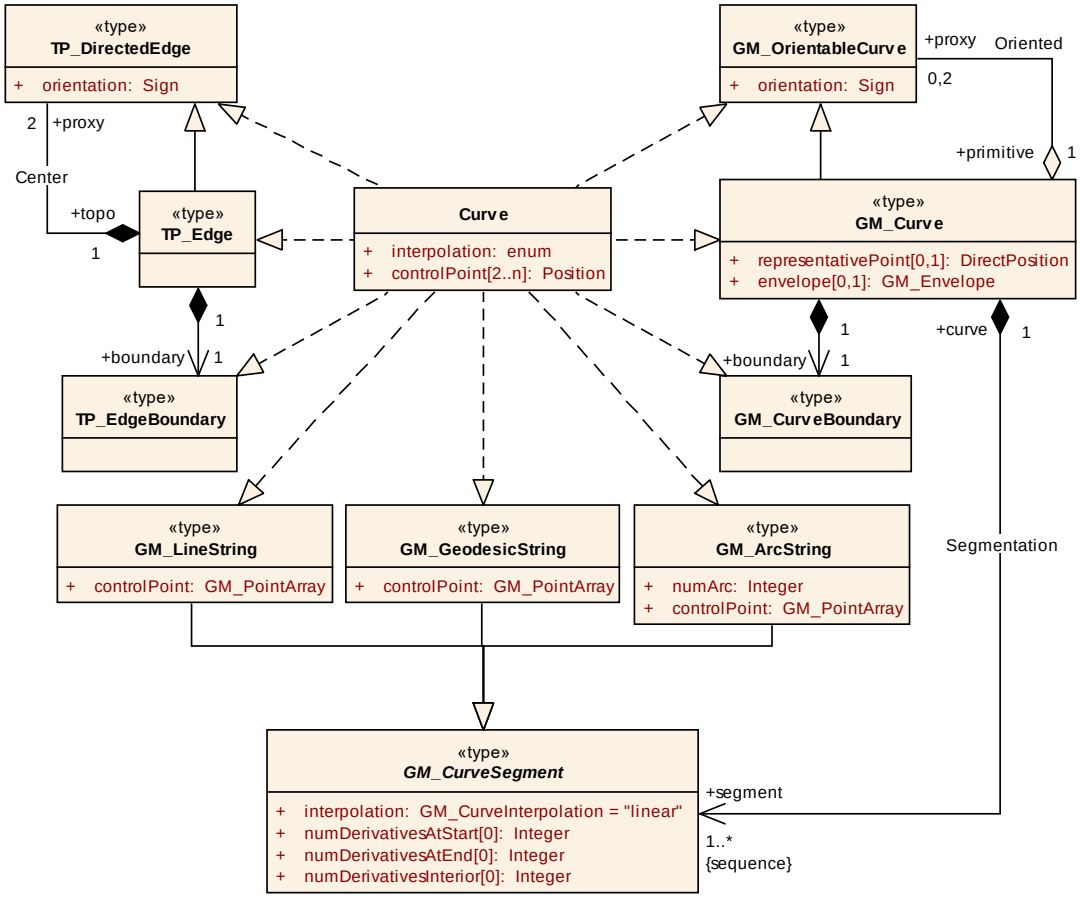


Figure K.59 – Curve realization

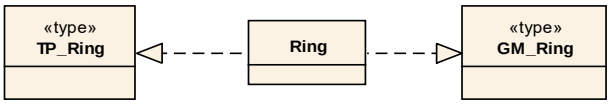


Figure K.60 – Ring realization

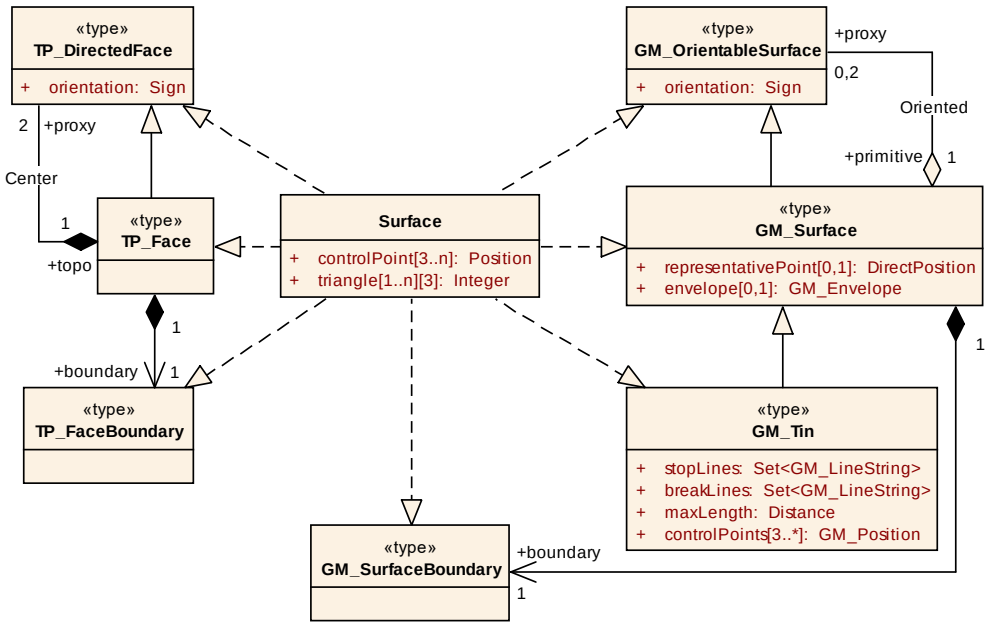


Figure K.61 – Surface realization

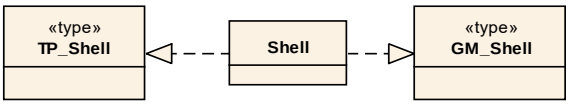


Figure K.62 – Shell realization

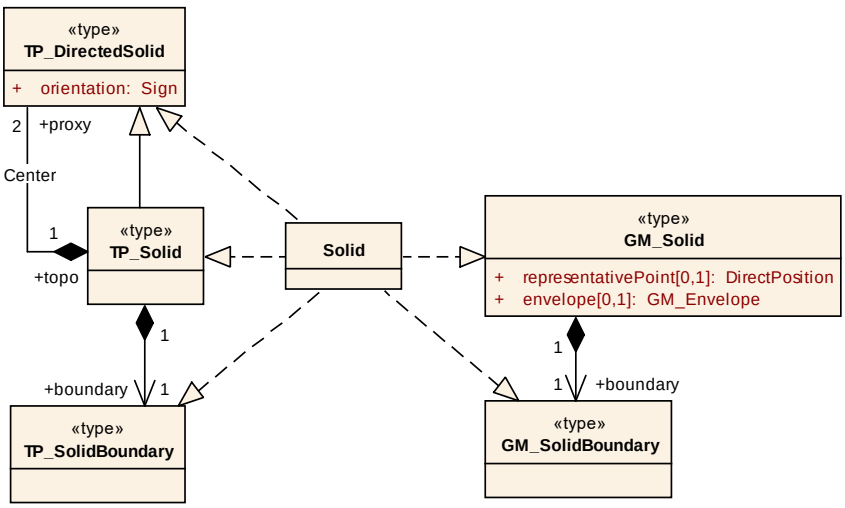


Figure K.63 – Solid realization

Annex L – Thoughts on profiling ISO 19107

L.1 Introduction

This annex is intended to offer some thoughts and material regarding the development of the DGIWG Profiles of ISO 19107.

L.2 Process from standard to data structure specification

L.2.1 Basic process

The goal or purpose of producing the DGIWG Profiles of ISO 19107 is to facilitate the development of standard interchange format specifications and implementations using the ISO/TC211 suite of standards. Some of these standards are conceptual and some of these standards are concrete.

To use the concrete standards the process appears clear. Either, use a standard as is, or use a subset as permitted by the standard.

For the standards that define concepts this process does not appear as clear. Figure L.1 is an attempt to depict the process of using conceptual standards. The process begins with a collection of conceptual standards that are designed to fit together. These standards define concepts, not implementations. From these standards profiles are created; in the simplest terms a profile is a subset of a standard. The next step is to put the concepts in the profiles together and realize them, still as concepts, in an application schema (→ what is an application schema?). The application schema is then realized as an interchange format specification – no longer merely conceptual.

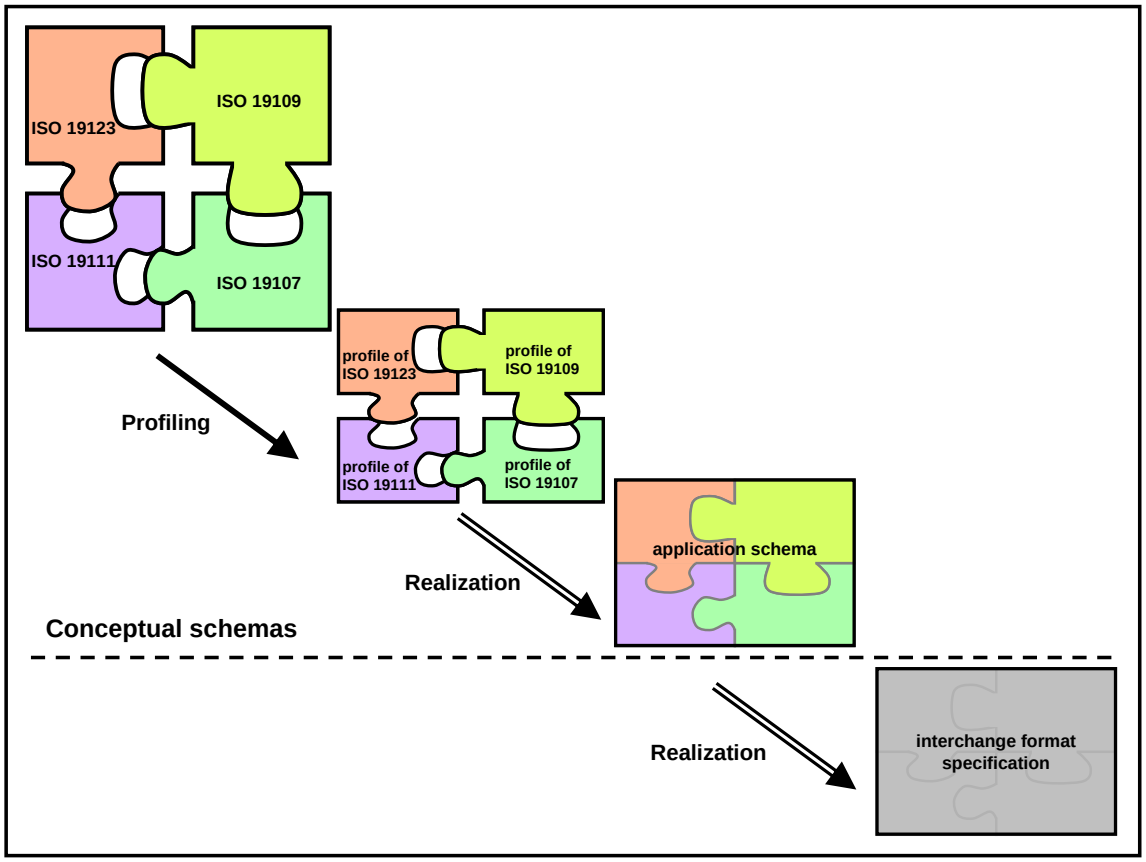


Figure L.1 – Process from standards to interchange format specification

The 19100 set of standards reference RM-ODP (ISO/IEC 10746) but this process does not appear to be explained there at all.

L.2.2 What is a profile?

The following are paragraphs taken from the 19100 collection of standards. They are included here to try and shed some light on the nature of profiles:

ISO 19101: § 6.1 Integration of geographic information with information technology: ¶ Profiles

Profiles and **functional standards** combine different standards in the ISO 19100 series and specialize the information in these standards in order to meet specific needs. **Profiles** and **functional standards** facilitate the development of geographic information systems and application systems that will be used for specific purposes. Clause 10 describes the approach to profiling the ISO 19100 series of standards.

ISO 19101: § 10.2 Profiles and base standards

A **profile** is a set of one or more **base standards** and, where applicable, the identification of chosen clauses, classes, subsets, options and parameters of those **base standards**, that is necessary to accomplish a particular function. A **base standard** is any standard in the ISO 19100 series or any other Information Technology standard that can be used as a source for components from which a **profile** may be constructed. **Base standards** define fundamentals and generalized procedures. They provide an infrastructure that can be used by a variety of applications, each of which constitutes a specific selection from the options offered.

ISO 19101: § 10.4 Use of Profiles

Profiles defining conforming subsets or combinations of the ISO 19100 series of base standards and/or subsets thereof, are used to perform specific functions. **Profiles** identify the use of particular options available in the **base standards**, and provide a basis for development of uniform, internationally recognized conformance tests.

ISO 19106: § 4.5

profile

set of one or more **base standards** or subsets of **base standards**, and, where applicable, the identification of chosen clauses, classes, options and parameters of those base standards, that are necessary for accomplishing a particular function [adapted from ISO/IEC TR 10000-1:1998]

NOTE A **profile** is derived from base standards so that by definition, conformance to a **profile** is conformance to the base standards from which it is derived.

L.2.3 What is an application schema?

The following are paragraphs taken from the 19100 collection of standards. They are included here to try and shed some light on the nature of application schemas:

ISO 19101: § 4.2

application schema

conceptual schema for data required by one or more applications

ISO 19106: § Introduction: ¶ 7

ISO 19109 defines the rules for the development of an **application schema**, including how the elements of **conceptual schemas** defined in other ISO geographic information standards are combined in an **application schema**. ISO 19109 guides the creation of **application schemas**, which is outside the scope of ISO 19106. An **application schema** by definition is not a **profile** but may integrate subsets of standardized schemas that are **profiles**.

L.2.4 Other terms

In the previous two sections other important terms were used and also there exist variants on the terms discussed above. Here is a list of these and other important terms:

- functional standards
- base standard
- conceptual schema
- implementation specification
- implementation profile

- transfer schema
- transfer standard

Most notable among these is the term *implementation profile*. How does this differ or is a specialization of *profile*? Two other terms that would require clarification are transfer schema and transfer standard. Some of the terms in the above list come from the following paragraphs:

ISO 19103: § 1 Scope: ¶ 3

One goal of the ISO 19100 series of standards using UML models is that they will provide a basis for mapping to ISO 19118, as well as a basis for creating **implementation specifications** for **implementation profiles** for various environments.

ISO 19107: § 5.1.5 Strong substitutability: ¶ 1

This International Standard assumes that **implementation profiles** and **transfer schemas** will be built using a strong version of substitutability. This means that at several places in designing an **application schema**, a profile builder may use a class in lieu of one defined in this schema as long as it supports the data, operation and associations required of the base class. The method of implementation of this substitutability is not normative, and may be done in a variety of manners depending on the characteristics of the implementation environment. This is especially true of **transfer standards**, which by their nature depend on data types. Entities in transfer sets may only be tenuously related to the base class in this International Standard, in that they may be data-only representational forms. Places where substitutability is most useful are examined in subclauses associated to the class most likely to take advantage of this technique.

ISO 19109: § Introduction: ¶ 4

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

L.2.5 The place of the profile in interchange

Does a profile define a maximum or a minimum? This depends on your perspective. Are you looking at the profile from the perspective of a data producer, or a data consumer. From the perspective of a data producer a profile defines a maximum. The data producer will only produce data with the classes, attributes, and associations specified in the profile. The data will be produced within the bounds of constrains, i.e. greater than or equal to minima and less than or equal to maxima. From the perspective of the data consumer the profile defines a minimum. The data consumer must be able to handle at least the classes, attributes, and associations specified in the profile. The data consumer at least will be able to handle multiplicities encompassing the bounds of constraints, i.e. less than or equal to minima and greater than or equal to maxima. This is analogous to the Design by Contract paradigm of Eiffel (ISO/IEC 25436). Figure L.2 is a graphical representation of this relationship.

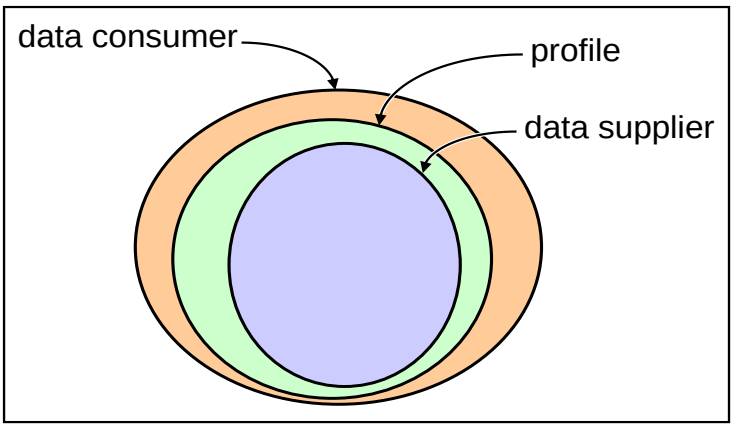


Figure L.2 – Profile conformance for data consumers and data suppliers

It is important to distinguish between flexibility in the profile and flexibility in the data. A profile may define a range for an association's multiplicity but the cardinality of a particular instance must be correct. For example, if a profile defines a boundary association, the multiplicity may have a range from 0..n. For a particular instance the relationship must reference all existing boundary objects not just some.

L.2.6 Profiling parts of a standard

Up to this point profiles have been presented as subsets of one or more base standards. Is it possible to create profiles of parts of one standard in order to be able to create combination profiles? For example, one could create profiles of GM primitives, of GM association containers, of GM complex containers, and of TP primitives. These profiles could be combined to create super-profiles thus avoiding the combinatorial dilemma of these orthogonal pieces.

L.2.7 Allowable range constraints

How may ranges be constrained? In "ISO 19137 - Generally used profiles of the spatial schema" the *Interior To* association of GM_Primitive and the *Contains* association of GM_Complex are constrained. In both case both roles are constrained to 0. The effect is that these associations are not used in this profile. In TP adjacency relationships are represented with the *Boundary* and *Coboundary* associations. Is it allowable to represent only half of an adjacency relationship? Because they are separated into two associations I think this is possible. In the case where the two directions of the adjacency relationship are joined into one association, as with *Isolated In*, it would be debatable. Because the reverse relationship, in this case, is easily derived it should be allowable.

L.3 A breakdown of ISO 19107

Table L.1 shows the classes of ISO 19107 organized into groups significant in profiling. Boxes are placed around the major groups. Classes not within a group are not significant for profiling. These are mostly classes that are either abstract root classes that cannot be excluded or auxiliary data types. Within each group there may be subgroups, most notably divided according to dimension.

Looking at the table the top group is the GM boundary group. This group has primitive (subgrouped by dimension) and complex boundary classes as well as classes for primitive boundary elements (GM_Ring and GM_Shell). The next group is the geometric primitive group. These too are subgrouped by dimension.

The third group is the segmentation group. This group is different in that for the profiling process segmentation classes may be chosen from a palette of possibilities.

The next two groups are GM container groups. The first is the unconstrained container group which contains GM_Aggregate and its multi-primitive subclasses subgrouped by dimension. The second is the constrained container group which contains GM_Complex and its composite primitive subclasses, which also are subgrouped by dimension. Note that what qualifies as a complex or composite also qualifies as an aggregate or multi primitive.

The five groups to this point have been GM group. The following groups are TP groups. The first of these is the TP primitive group. This group contains both boundary and primitive classes. This is in contrast with GM where boundary and primitive classes are in separate groups. The reason behind this is that GM primitive classes can be in a profile without GM boundary classes, whereas since TP primitives are required to be in a complex they should always occur together with TP boundary classes.

The final group is the expression group. This is not significant for us in the context of an interchange format

Geometry Package	0D	1D	2D	3D
Root				
GM_Object				
Geometry				
Boundary				
GM_Boundary				
GM_PrimitiveBoundary		GM_CurveBoundary	GM_SurfaceBoundary	GM_SolidBoundary
GM_ComplexBoundary				
Closed Boundary Elements				
			GM_Ring	GM_Shell
Primitives				
GM_Primitive	GM_Point	GM_Curve	GM_Surface	GM_Solid
GM_OrientablePrimitive		GM_OrientableCurve	GM_OrientableSurface	
Generic Prmitives				
		GM_GenericCurve	GM_GenericSurface	
Position				
Bearing				
DirectPosition				
TransfiniteSet<DirectPosition>				
GM_PointRef				
GM_Envelope				
GM_Position		GM_PointArray	GM_PointGrid	
Segmentation				
Segmentation Root				
		GM_CurveSegment	GM_SurfacePatch	
		GM_CurveInterpolation	GM_SurfaceInterpolation	
Segmentation				
		GM_LineString	GM_PolyhedralSurface	
		GM_LineSegment	GM_Polygon	
		GM_GeodesicString	GM_TriangulatedSurface	
		GM_Geodesic	GM_Triangle	
		GM_ArcString	GM_Tin	
		GM_Arc	GM_ParametricCurveSurface	
		GM_Circle	GM_GriddedSurface	
		GM_ArcStringByBulge	GM_Cone	
		GM_ArcByBulge	GM_Cylinder	
		GM_Conic	GM_Sphere	
		GM_Placement	GM_BilinearGrid	
		GM_AffinePlacement	GM_BicubicGrid	
		GM_Clothoid	GM_BSplineSurfaceForm	
		GM_OffsetCurve	GM_BSplineSurface	
		GM_Knot		
		GM_KnotType		
		GM_SplineCurve		
		GM_PolynomialSpline		
		GM_CubicSpline		
		GM_SplineCurveForm		
		GM_BSplineCurve		
		GM_Bezier		
Containers				
Unconstrained containers				
GM_Aggregate				
GM_MultiPrimitive	GM_MultiPoint	GM_MultiCurve	GM_MultiSurface	GM_MultiSolid
Constrained containers				
GM_Complex				
GM_Composite	GM_CompositePoint	GM_CompositeCurve	GM_CompositeSurface	GM_CompositeSolid
Topology Package				
Root				
TP_Object				
Topology				
Boundary				
TP_Boundary				
TP_PrimitiveBoundary		TP_EdgeBoundary	TP_FaceBoundary	TP_SolidBoundary
TP_ComplexBoundary				
Closed Boundary Elements				
			TP_Ring	TP_Shell
Primitives				
TP_Primitive	TP_Node	TP_Edge	TP_Face	TP_Solid
TP_DirectedTopo	TP_DirectedNode	TP_DirectedEdge	TP_DirectedFace	TP_DirectedSolid
Containers				
TP_Complex				
Expressions				
TP_Expression				
TP_ExpressionTerm				

Table L.1 – Classes of ISO 19107 organized into groups significant in profiling

Figure L.3, below attempts to divide the space of ISO 19107 into profiling units. The groupings in Table L.1 are reflected in profiling units of Figure L.3. This figure does not distinguish between GM and TP. It treats TP primitives merely as an extension of GM primitives. The striping in the spatial primitive units and container units indicates dimension specific subunits.

The division of spatial primitives into primitive geometry, boundary geometry, and topology does not completely reflect the grouping in the table. The boundary geometry unit at one level does not require boundary class but just includes the fact that all boundary objects are present, forming a complex. At another level the boundary geometry unit encompasses the inclusion of boundary classes as well.

The segmentation unit has both vertical and horizontal striping. The vertical striping indicates the dimension specific nature of its subunits. The horizontal striping indicates the palette nature of the dimension specific subunits.

Regarding containers, TP_Complex replicates GM_Complex and is folded into the constrained container unit. Here again there is vertical striping to indicate dimension specific subunits. The arrows indicate to which level of spatial primitives the containers apply. Aggregations and multi-primitives may apply to any type of primitive. Complexes and composites may only apply to primitives within a complex and this requires boundary primitives to exist.

Expressions are topology specific but of no interest to us in the context of an interchange format.

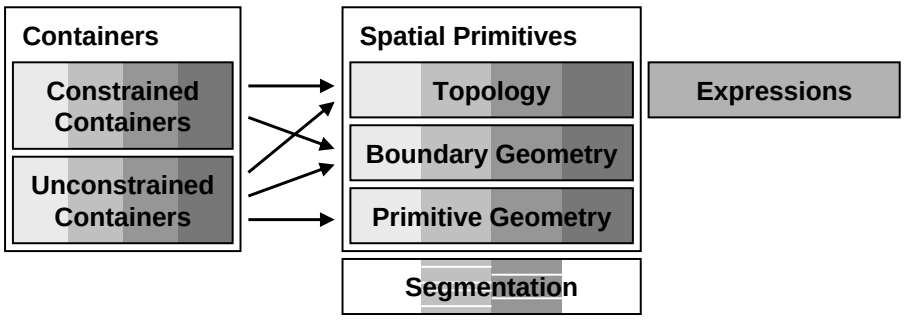


Figure L.3 – ISO 19107 divided into profiling units

The units in Figure L.3 can be used as axes for defining the multidimensional profile space. In the following figures these axes will be selected and graphed against each other.

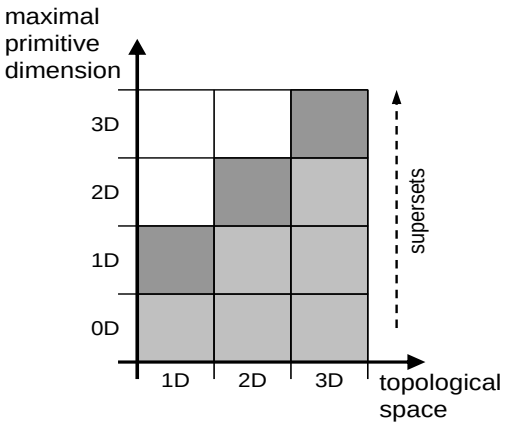


Figure L.4 – Profile space defined by topological space and primitive dimension

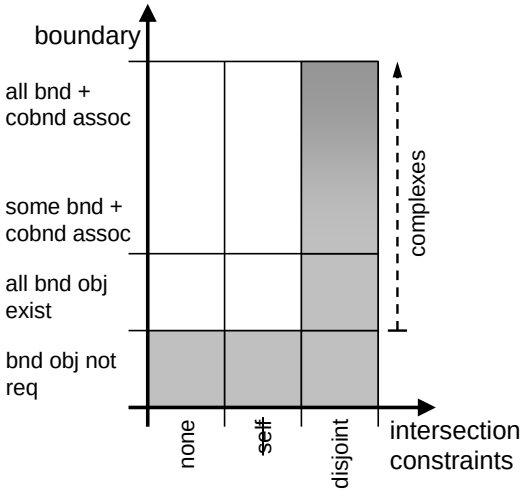


Figure L.5 – Profile space defined by primitive intersection constraints and boundary associations

Figure L.4 shows the axes of dimensionality. The horizontal axis defines the topological space. A 1D coordinate space is possible but of no interest to us. The profiles in the 2D topological space are defined in the 2D and 2½D profiles document while the profiles in the 3D topological space are defined in the 3D profiles document. Note that 2½D refers to a 2D topological space combined with a 3D coordinate space. The vertical axis defines the maximal primitive dimension. Since the highest dimension profile in each column is a superset of all profiles beneath it, it is only necessary to specify the top profile (in dark gray) and allow profiles of lesser maximal dimension to be defined as subprofiles of the top profile.

Figure L.5 graphs levels of boundary objects and boundary associations against intersection constraints. At the bottom level on the vertical, “boundary”, axis boundary objects are not required. That is to say, for any primitive its bounding primitives may be absent. At the next level all boundary objects exist but there are no associations binding a primitive to its boundary. Therefore, for every primitive the primitives that bound it exist. The next level is a graduated level with increasing boundary and coboundary associations. The horizontal axis defines levels of intersection constraints. At the first level there are no intersection constraints. At the second level self intersection is not permitted but intersection between distinct primitives is permitted. At the third level all primitives are disjoint. Note that in this graph the boundary level does not increase until the maximal intersection constraint is reached. Because of this, the graph can be projected onto one axis. It might be difficult to name the axis, but ultimately it is an axis of increasing complexity. Also note that the profiles, where all primitives are disjoint and all boundary objects exist, define complexes.

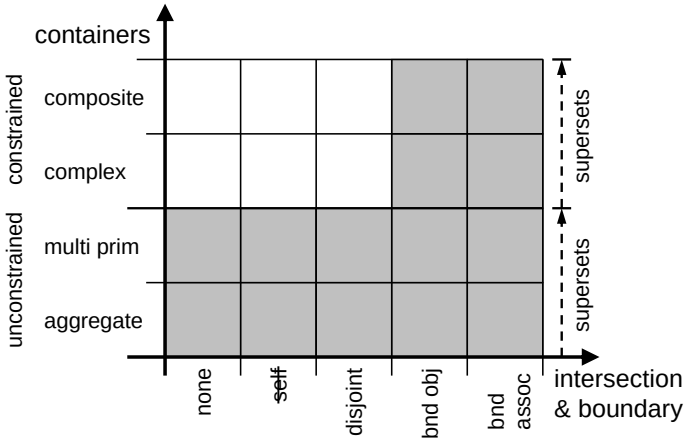


Figure L.6 – Profile space defined by intersection/boundary constraint and container complexity

Figure L.6 shows which containers are possible at which level of complexity. Aggregates and multi-primitives, as unconstrained containers, are always available. Complexes and composites, on the other hand, can only be use where the primitives form a complex.

L.4 Profiling boundaries

L.4.1 Introduction

As is clear from Figure L.5, there is a continuum of levels of complexity relative to boundary associations from none at all to highly structured bidirectional associations. This section will discuss this continuum in more detail. The first subsection will discuss the various levels of boundary association. The second subsection will discuss the effort needed to derive additional information from the levels of varying complexity.

L.4.2 Levels of boundary association

L.4.2.1 Introduction

The following sections show examples of how the boundary relationships could be profiled. In the examples primitives are designated with both their GM and TP terms. The focus is on the associations between surfaces/faces and curves/edges. Pairs of directed primitives are merged into one primitive. Only primitives in a complex are considered.

In ISO 19107 the TP *boundary* and *coboundary* are specified as operations, associations, and derived associations. Because of this it is not clear to what degree these associations must be realized to conform to ISO 19107. The associations shown are ones that are actually realized as opposed to merely derived.

L.4.2.2 Complex with no associations

This is the simplest level. There is a complex, i.e. all boundary objects exist and all primitives are disjoint, but there are no associations between the primitives. These are only GM objects.

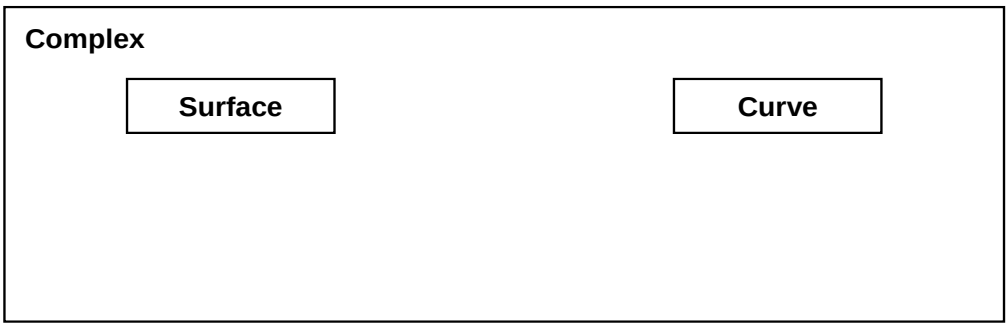


Figure L.7 – Complex with no associations between the primitives

L.4.2.3 Adding the boundary operation

This example adds the *boundary* operation to the previous example. There are still no associations between the primitives. Both GM and TP specify a *boundary* operation. The two packages parallel each other in the structure of their boundary objects as well.

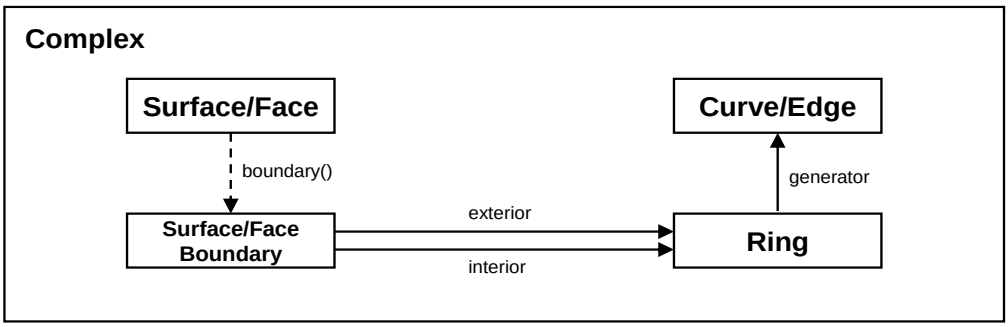


Figure L.8 – Complex with boundary operation but no explicit boundary association

L.4.2.4 Operation boundary as association

In this example *boundary* is realized as an association. GM does not explicitly allow the boundary operation to be realized as an association but this usage is clearly implied in ISO 19107 § 5.1.5 Strong substitutability ¶ 1. In TP there is both a *boundary* operation and a *boundary* association. The *boundary* operation returns a boundary object whereas the *boundary* association associates directly with the boundary primitives.

Although the association here is not identical to the TP *boundary* it does express the intent and is therefore acceptable under the principle of strong substitutability.

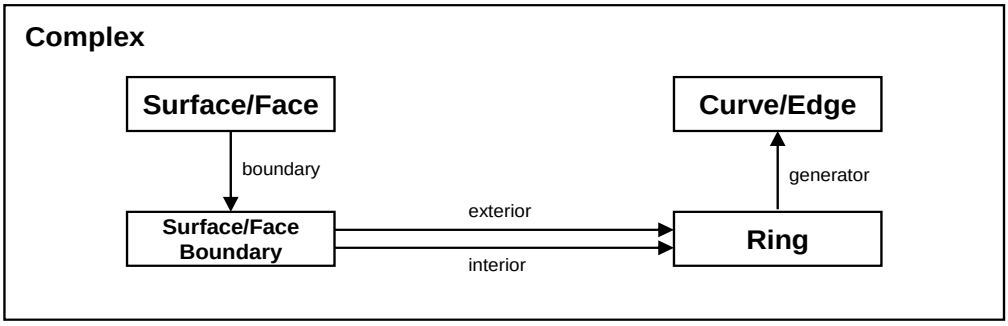


Figure L.9 – Complex with boundary operation implemented as an association

L.4.2.5 Operations boundary and coboundary

In this example both operations *boundary* and *coboundary* are included in the profile. There are still no associations between the primitives. Both GM and TP specify a *boundary* operation while *coboundary* is strictly a TP operation.

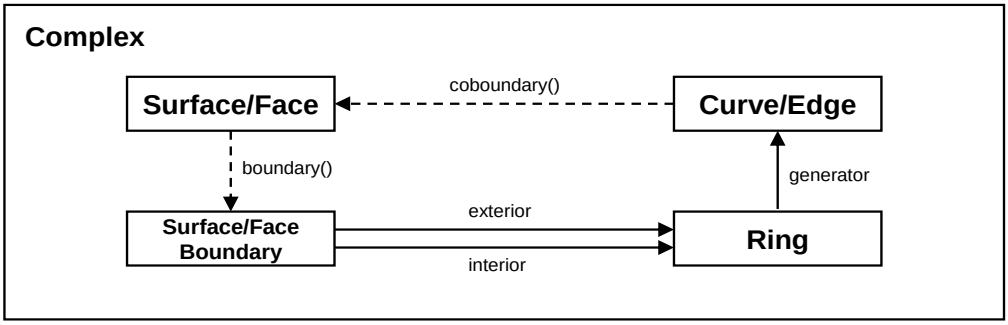


Figure L.10 – Complex with boundary and coboundary operations

L.4.2.6 Operations "boundary" and "coboundary" as associations

In this example *boundary* and *coboundary* are realized as associations. This is a variant defined only in TP.

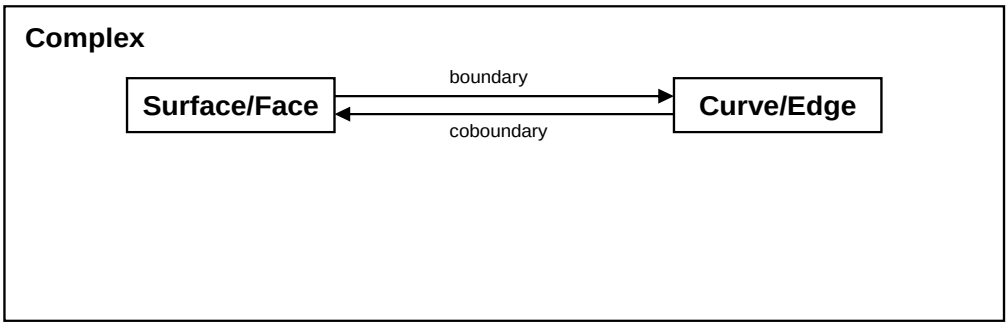


Figure L.11 – Topological boundary and coboundary associations

L.4.2.7 Data only with full topological relationships

In this example both *boundary* and *coboundary* are realized as associations. Here again GM does not explicitly allow the boundary operation to be realized as an association but this usage is clearly implied in ISO 19107 § 5.1.5 Strong substitutability ¶ 1. In TP there is both a *boundary* operation and a *boundary* association. The *boundary* operation returns a boundary object whereas the *boundary* association associates directly with the bounding primitives. Although the association here is not identical to the TP *boundary* it does express the intent and is therefore acceptable under the principle of strong substitutability. *coboundary* is strictly TP and this representation is a variant as specified.

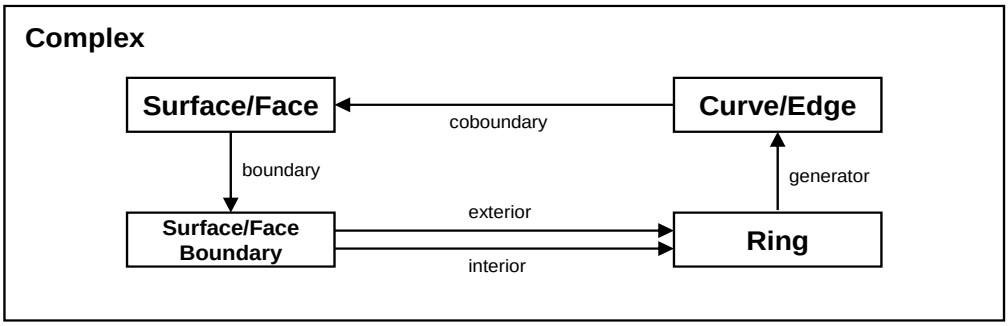


Figure L.12 – Boundary associations with boundary objects

L.4.2.8 Realization

In a realization of the profile in the previous section the Surface/Face class can be merged with the Surface/Face Boundary class resulting in the structure shown in Figure L.13. This realizes GM and TP objects. The *boundary* operation/association is realized by the *exterior* and *interior* associations to the Ring and the *generator* association to the Curve/Edge. This is not a literal realization but is acceptable under the principle of strong substitutability.

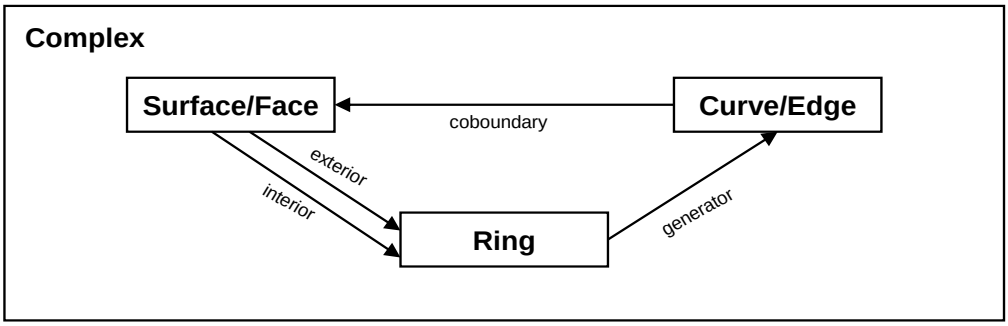


Figure L.13 – Realization of geometric and topological primitives

L.4.3 Storing versus deriving association information in a complex

L.4.3.1 Introduction

As show in the previous section, given a complex, there are varying degrees to which adjacency relationships may be stored in associations versus derived in operations. This section discusses the effort required to derived relationships that could otherwise be stored in associations. Of all the steps required spatial searches require the greatest effort.

L.4.3.2 Deriving 2D boundary information

Introduction

Figure L.14 shows the most information rich adjacency relationships between primitives in a 2D topological space. Each following subsection will discuss the steps necessary to reach this level of information from a given level of association.

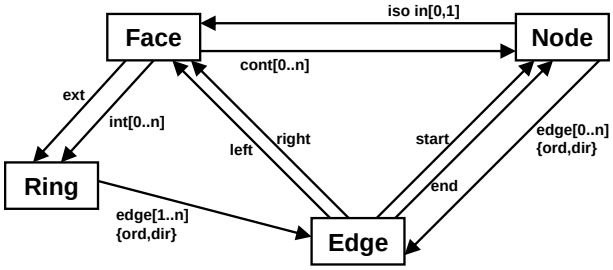


Figure L.14 – Fully realized containment relationships in 2 dimensions

Each section will list the directed relationships between the primitives. For example the Edge – Node relationship will be indicated as "E→N". Following this the steps to achieve the end result will be discussed. Steps that are marked '*' are not required if all of a specified type of association are being established, then one pass is performed to establish all of the relationships. A '→' indicates the end result.

Only primitives in complex

This first example is the simplest. The primitives in the complex have no explicit adjacency associations.

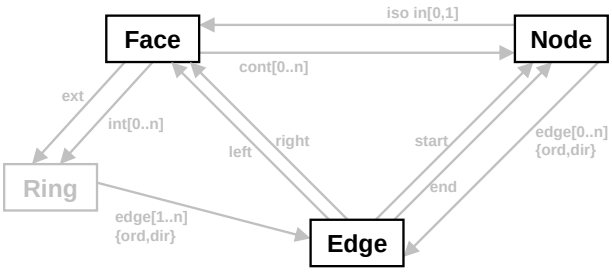


Figure L.15 – Unassociated complex

- E➔N: perform spatial search to find start node and end node
  → start node and end node
- N➔E: perform spatial search to find cobounding edges
  establish orientation of edges around node
  sort edges around node
  → ordered list of directed edges
- F➔E: perform spatial search to find bounding edges
  group, sort, and orient edges around face
  → ordered rings of directed edges
- E➔F: perform spatial search to find cobounding faces
  test side of faces relative to edge
  → left and right directed faces
- F➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔F: perform spatial search to find containing face
  → containing face

Only simple boundary associations

This example uses the simple boundary association as defined in TP. A Face points at its bounding directed Edges but no Rings are defined.

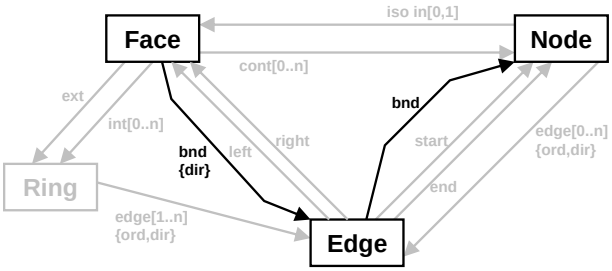


Figure L.16 – Simple boundary associations

- E➔N: test collocation to determine which is start and which is end node
  → start node and end node
- N➔E: *perform search, linear or spatial, to find cobounding edges
  establish orientation of edges around node

sort edges around node

→ ordered list of directed edges

- F➔E: group, sort, and orient edges around face
  → ordered rings of directed edges
- E➔F: *perform search, linear or spatial, to find cobounding faces
  test side of faces relative to edge
  → left and right directed faces
- F➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔F: perform spatial search to find containing face
  → containing face

Only simple boundary and coboundary associations

This example uses the simple boundary and coboundary associations as defined in TP. A Face points at its bounding directed Edges but no Rings are defined. A Node points at its cobounding directed Edges but no order is defined.

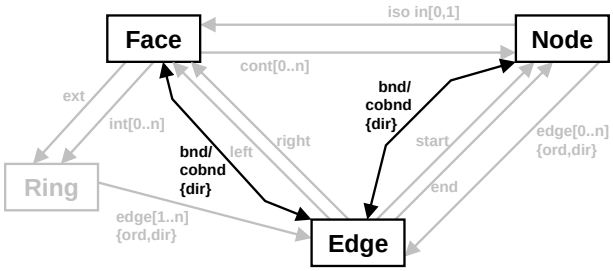


Figure L.17 – Simple boundary and coboundary associations

- E➔N: test collocation to determine which is start and which is end node
  → start node and end node
- N➔E: sort edges around node
  → ordered list of directed edges
- F➔E: group and sort edges around face
  → ordered rings of directed edges
- E➔F: use orientation of face to determine side relative to edge
  → left and right directed faces
- F➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔F: perform spatial search to find containing face
  → containing face

Full boundary associations

In this example the full boundary associations between primitives are stored.

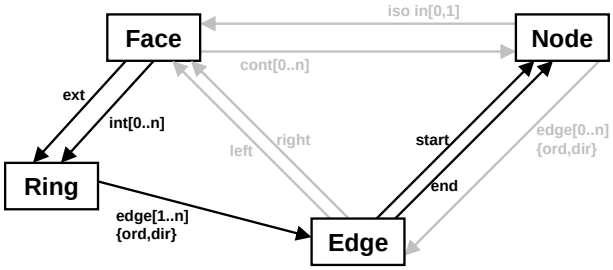


Figure L.18 – Full boundary associations

- E➔N: → start and end nodes
- N➔E: *perform search, linear or spatial, to find cobounding edges
  establish orientation of edges around node
  sort edges around node
  → ordered list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: *perform search, linear or spatial, to find cobounding faces
  test side of faces relative to edge
  → left and right directed faces
- F➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔F: perform spatial search to find containing face
  → containing face

Full boundary and containment structures
In this example the full boundary and containment associations between primitives are stored.

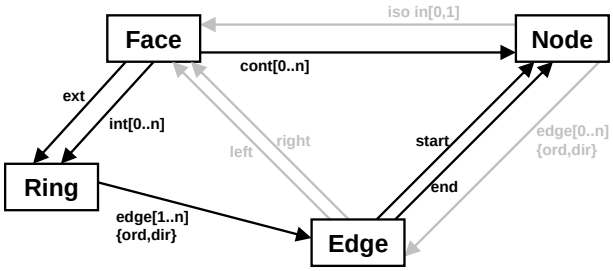


Figure L.19 – Full boundary and containment associations

- E➔N: → start and end nodes
- N➔E: *perform search, linear or spatial, to find cobounding edges
  establish orientation of edges around node
  sort edges around node
  → ordered list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: *perform search, linear or spatial, to find cobounding faces
  test side of faces relative to edge
  → left and right directed faces

- F➔N: → interior nodes
- N➔F: * perform search, linear or spatial, to find containing face
  → containing face

Full boundary, coboundary and containment

In this example the full boundary, coboundary and containment associations between primitives are stored.

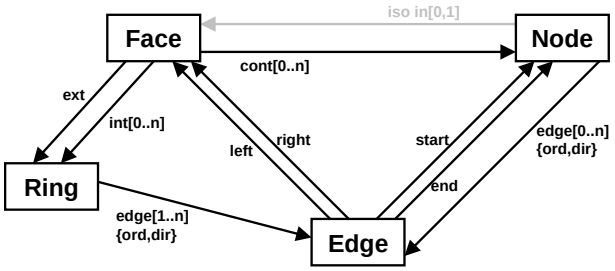


Figure L.20 – Full boundary, coboundary and containment associations

- E➔N: → start and end nodes
- N➔E: → ordered list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: → left and right directed faces
- F➔N: → interior nodes
- N➔F: * perform search, linear or spatial, to find containing face
  → containing face

Full boundary, coboundary, containment and isolated in

In this example all boundary, coboundary, containment and isolated in associations between primitives are stored. No effort is necessary to derive relationships.

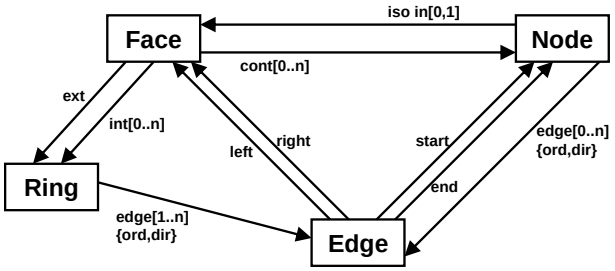


Figure L.21 – Full boundary, coboundary, containment and isolated in associations

- E➔N: → start and end nodes
- N➔E: → ordered list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: → left and right directed faces
- F➔N: → interior nodes
- N➔F: → containing face

L.4.3.3 Deriving 3D boundary information

Introduction

Figure L.22 shows the most information rich adjacency relationships between primitives in a 3D topological space. Each following subsection will discuss the steps necessary to reach this level of information from the given level of association.

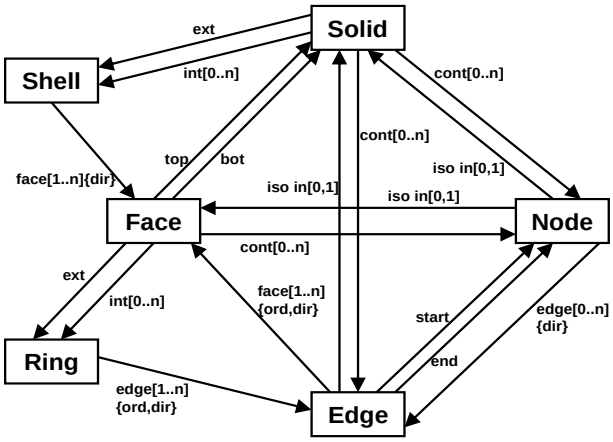


Figure L.22 – Fully realized containment relationships in 2 dimensions

Each section will list the directed relationships between the primitives. For example the Edge – Node relationship will be indicated as "E➔N". Following this the steps to achieve the end result will be discussed. Steps that are marked '*' are not required if all of a specified type of association are being established, then one pass is performed to establish all of the relationships. A '➔' indicates the end result.

Only primitives in complex

This first example is the simplest. The primitives in the complex have no explicit adjacency associations. There are no Solids included here because Solids have no segmentation representation in ISO 19107. To represent a Solid a Solid must be related to its bounding Faces.

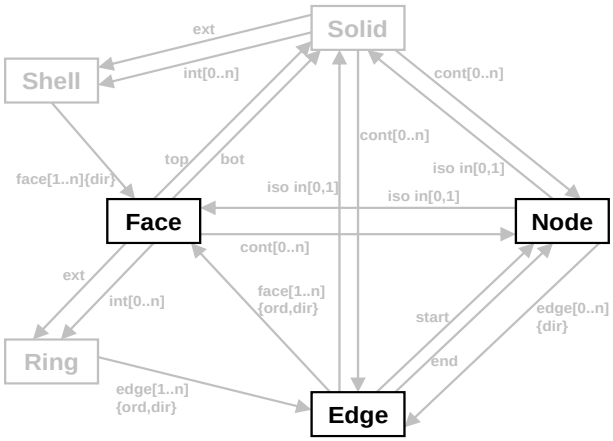


Figure L.23 – Unassociated complex

- E➔N: perform spatial search to find start node and end node

  ➔ start node and end node

- N➔E: perform spatial search to find cobounding edges

  establish orientation of edges around node

  ➔ list of directed edges

- F➔E: perform spatial search to find bounding edges

  group, sort, and orient edges around face

  ➔ ordered rings of directed edges

- E➔F: perform spatial search to find cobounding faces

  establish orientation of faces around edge

  sort faces around edge

  ➔ ordered list of directed faces

- S➔F (no solids in this profile)
- F➔S (no solids in this profile)
- F➔N: perform spatial search to find interior nodes

  → interior nodes
- N➔F: perform spatial search to find containing face

  → containing face
- S➔N (no solids in this profile)
- N➔S (no solids in this profile)
- S➔E (no solids in this profile)
- E➔S (no solids in this profile)

Only simple boundary associations

This example uses the simple boundary association as defined in TP. A Face points at its bounding directed Edges but no Rings are defined. A Solid points at its bounding Faces but no Shells are defined.

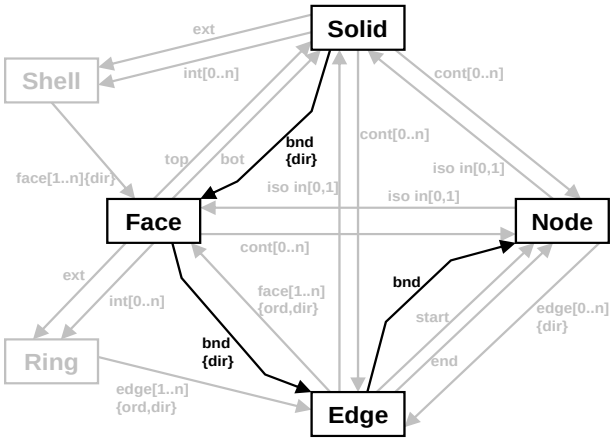


Figure L.24 – Simple boundary associations

- E➔N: test collocation to determine which is start and which is end node

  → start node and end node
- N➔E: *perform search, linear or spatial, to find cobounding edges

  establish orientation of edges around node

  → list of directed edges
- F➔E: group, sort, and orient edges around face

  → ordered rings of directed edges
- E➔F: *perform search, linear or spatial, to find cobounding faces

  establish orientation of faces around edge

  sort faces around edge

  → ordered list of directed faces
- S➔F: group and orient faces in shells

  → shells of directed faces
- F➔S: *perform search, linear or spatial, to find cobounding solids

  test side of solids relative to face

  → top and bottom directed solid

- F➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔F: perform spatial search to find containing face
  → containing face
- S➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔S: perform spatial search to find containing solid
  → containing solid
- S➔E: perform spatial search to find interior edges
  → interior edges
- E➔S: perform spatial search to find containing solid
  → containing solid

Only simple boundary and coboundary associations

This example uses the simple boundary and coboundary associations as defined in TP. A Face points at its bounding directed Edges but no Rings are defined. A Solid points at its bounding Faces but no Shells are defined. An Edge points at its cobounding directed Faces but no order is defined.

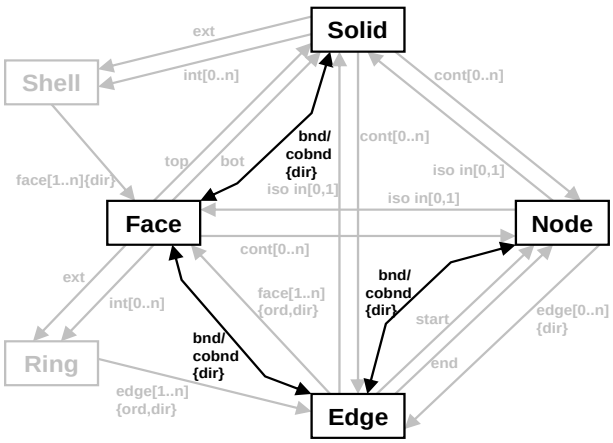


Figure L.25 – Simple boundary and coboundary associations

- E➔N: test collocation to determine which is start and which is end node
  → start node and end node
- N➔E: → list of directed edges
- F➔E: group and sort edges around face
  → ordered rings of directed edges
- E➔F: sort faces around edge
  → ordered list of directed faces
- S➔F: group faces in shells
  → shells of directed faces
- F➔S: test orientation of solids relative to face
  → top and bottom directed solid
- F➔N: perform spatial search to find interior nodes
  → interior nodes

- N➔F: perform spatial search to find containing face
  → containing face
- S➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔S: perform spatial search to find containing solid
  → containing solid
- S➔E: perform spatial search to find interior edges
  → interior edges
- E➔S: perform spatial search to find containing solid
  → containing solid

Full boundary structures
In this example the full boundary associations between primitives are stored.

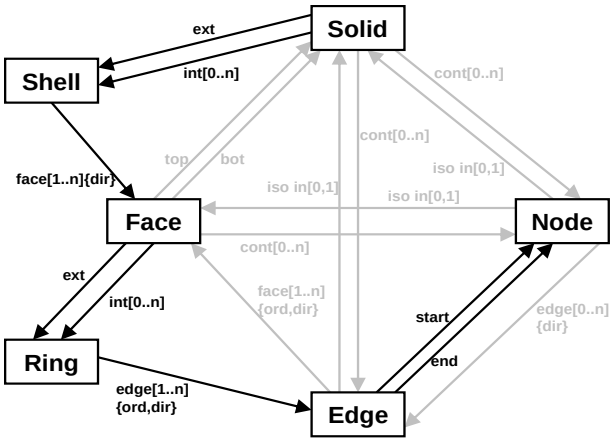


Figure L.26 – Full boundary associations

- E➔N: → start node and end node
- N➔E: *perform search, linear or spatial, to find cobounding edges
  establish orientation of edges around node
  → list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: *perform search, linear or spatial, to find cobounding faces
  establish orientation of faces around edge
  sort faces around edge
  → ordered list of directed faces
- S➔F: → shells of directed faces
- F➔S: *perform search, linear or spatial, to find cobounding solids
  test side of solids relative to face
  → top and bottom directed solid
- F➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔F: perform spatial search to find containing face

→ containing face

- S➔N: perform spatial search to find interior nodes
  → interior nodes
- N➔S: perform spatial search to find containing solid
  → containing solid
- S➔E: perform spatial search to find interior edges
  → interior edges
- E➔S: perform spatial search to find containing solid
  → containing solid

Full boundary and containment structures
In this example the full boundary and containment associations between primitives are stored.

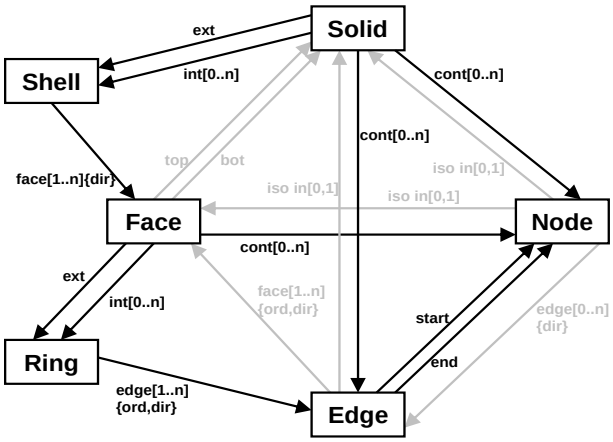


Figure L.27 – Full boundary and containment associations

- E➔N: → start node and end node
- N➔E: *perform search, linear or spatial, to find cobounding edges
  establish orientation of edges around node
  → list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: *perform search, linear or spatial, to find cobounding faces
  establish orientation of faces around edge
  sort faces around edge
  → ordered list of directed faces
- S➔F: → shells of directed faces
- F➔S: *perform search, linear or spatial, to find cobounding solids
  test side of solids relative to face
  → top and bottom directed solid
- F➔N: → interior nodes
- N➔F: * perform search, linear or spatial, to find containing face
  → containing face
- S➔N: → interior nodes

- N➔S: *perform spatial search to find containing solid
  → containing solid
- S➔E: → interior edges
- E➔S: *perform spatial search to find containing solid
  → containing solid

Full boundary, coboundary and containment
In this example the full boundary, coboundary and containment associations between primitives are stored.

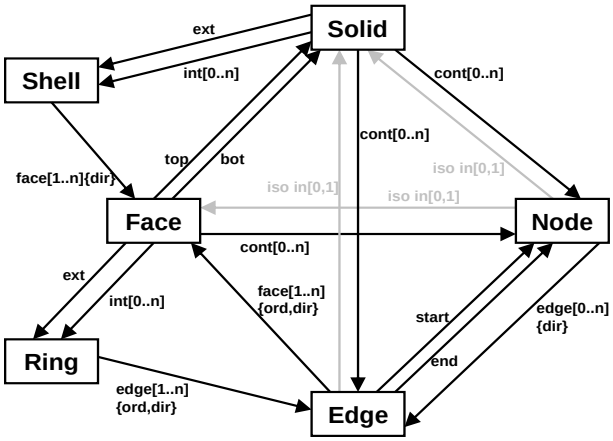


Figure L.28 – Full boundary, coboundary and containment associations

- E➔N: → start node and end node
- N➔E: → list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: → ordered list of directed faces
- S➔F: → shells of directed faces
- F➔S: → top and bottom directed solid
- F➔N: → interior nodes
- N➔F: * perform search, linear or spatial, to find containing face
  → containing face
- S➔N: → interior nodes
- N➔S: *perform spatial search to find containing solid
  → containing solid
- S➔E: → interior edges
- E➔S: *perform spatial search to find containing solid
  → containing solid

L.4.3.4 Full boundary, coboundary, containment and isolated in

In this example all boundary, coboundary, containment and isolated in associations between primitives are stored. No effort is necessary to derive relationships.

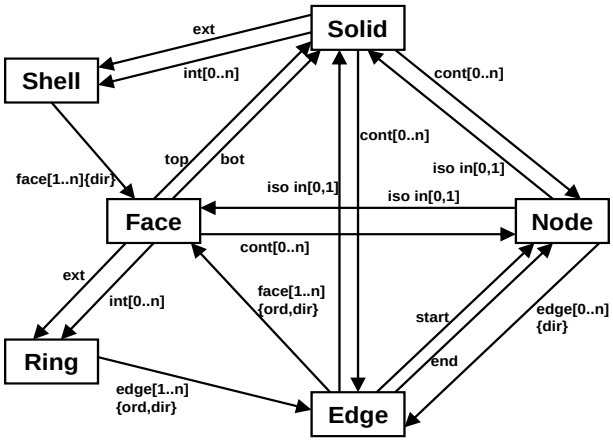


Figure L.29 – Full boundary, coboundary, containment and isolated in associations

- E➔N: → start node and end node
- N➔E: → list of directed edges
- F➔E: → ordered rings of directed edges
- E➔F: → ordered list of directed faces
- S➔F: → shells of directed faces
- F➔S: → top and bottom directed solid
- F➔N: → interior nodes
- N➔F: → containing face
- S➔N: → interior nodes
- N➔S: → containing solid
- S➔E: → interior edges
- E➔S: → containing solid

L.4.3.5 Operations for deriving adjacency relationships

From the examples we know that there are four different operations that are performed to derive adjacency relationships in complexes:

- spatial search
- orientation
- grouping
- ordering

Of these spatial searches certainly require the greatest effort. Orientation is part of the data structure, therefore if a relationship with orientation information exists in one direction then the orientation is implied in the other direction. Grouping is in regard to forming Rings and Shells. Ordering is for Edges around Nodes in 2D, Edges around Rings, and Faces around Edges in 3D. Faces around Edges and Solids around Faces both in 3D are minor forms of orientation and ordering.

L.5 Conclusion: levels of profiles

From section 2 it is clear that there are three primary axes for creating profiles of ISO 19107: dimension, available containers, and intersection & boundary. Of these three axes intersection & boundary is most complex. And of this axis the boundary part is more complex still. Containers are the simplest and can be included in all profiles. Of the dimension axis the highest dimension profile can be documented and lower dimension profiles specified as subprofiles. This leaves the major profile levels stemming from the intersection & boundary axis. They are the following:

1. Geometry only, spaghetti
   This profile has problems with solids. Solids can only be represented by their boundaries

in ISO 19107, where otherwise this profile requires no boundaries and has no boundary associations.

2. Geometry only, no self intersection
   This profile has the same problem with solids as the previous profile.
3. Geometry only, primitives disjoint
   This profile also has the problem with solids.
4. Complex, no boundary associations
   This profile has the problem with solids in a diminished form; only the boundary associations are missing.
5. Full adjacency associations

Between the last two of the five profiles above there is the profile continuum where many different profiles could be inserted. From profile 4 the next step up would be to add solids. This would be a complex with only the boundary associations needed to make solids. From profile 5 a step down would be to remove clearly redundant information. This would be in particular the *IsolatedIn* association; given one direction of an IsolatedIn association implies the reverse. In order to avoid a variable length list using the association to the container is simpler having a multiplicity of 0 or 1. Between these two profiles it might be desirable to have one or two more profile, but this will be left open for discussion. The list of profiles is then:

1. Geometry only, spaghetti
2. Geometry only, no self intersection
3. Geometry only, primitives disjoint
4. Complex, no boundary associations
5. Complex, no boundary associations + solids with boundary associations
6. Full adjacency associations, minus IsolatedIn redundancy
7. Full adjacency associations

Bibliography

A Bibliography may be included in a document. It contains references to additional material that is useful for the understanding of the document. The references to the material necessary for understanding the document and which by reference forms a part of the standard or standard operating procedure is included in clause 3 Normative references.

Examples related to this document

[1] ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*, 2001

[2] ISO/IEC TR 10000-1, *Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework*