DGIWG – 100
DGIWG 2D Spatial Schema Profile

Document Identifier:	STD-DP-03-080-ed1.0.1-2D_Spatial_Schema_Profile
Publication Date:	07 August 2003
Edition:	1.0.1
Edition Date:	07 August 2003
Responsible Party:	DGIWG
Audience:	Approved for public release
Abstract:	This is a subset of ISO 19107 classes (clause 6 and 7) which is the minimum required to support a 2 dimensional and 2.5 dimensional spatial schema. The profile was conducted with the support and collaboration of the International Hydrographic Organization (IHO), which provided insight and guidance to maritime needs and requirements.
Copyright:	

Contents

Figures

Constraints

Introduction

ISO 19107 “Geographical Information - Spatial schema” contains all the information necessary for describing and manipulating the spatial characteristics of geographical features. This information is divided into of a number of classes.

The spatial requirements of DIGEST – The Next Generation and S-57 Edition 4.0 are less comprehensive than the requirements of ISO 19107. This profile contains the subset of ISO 19107 classes which are used by DIGEST and S-57. Where possible, the language used by ISO 19107 has been simplified so that it is more understandable to non-specialists. A brief explanatory statement accompanies each class. For a full explanation, reference must be made to ISO 19107.

A key characteristic of spatial schema is the existence of a surface, 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, often referred to as a “manifold”, this profile supports 2-D and some 2.5D models. A 2.5D model is a 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 2.5D model surface primitives are fully defined in x and y but their interiors may be left undefined in z. This profile only supports 2.5D models in which the surface primitive interiors are undefined in z. Examples diagrams can be found in Annex A.

The profile is in two main parts dealing with geometry (Clause 6) and topology (Clause 7).

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.

1 Scope

This profile specifies:

a) a subset of ISO 19107 classes (clause 6 and 7) which is the minimum required to support a 2 dimensional and 2.5 dimensional spatial schema. As such it is restricted to only specifying data and does not include operations.

b) additional constraints (omitted optional elements or constrained cardinalities) which are imposed on these classes by this profile.

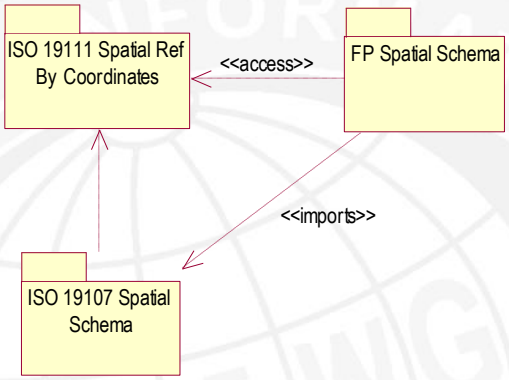


Figure 1 – FP Spatial Schema relationship with ISO19100 Packages

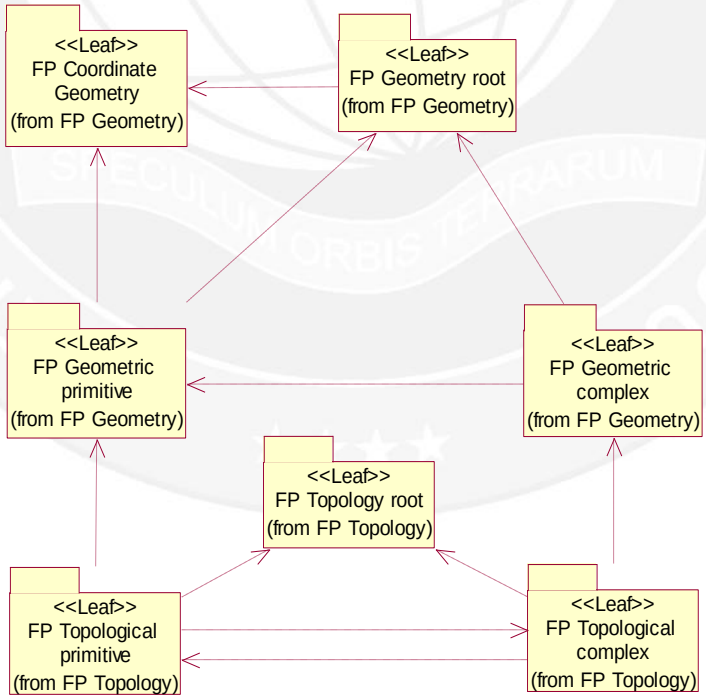


Figure 2 – FP Packages and their dependencies

2 Conformance

Clauses 6 and 7 of this profile use the Unified Modeling Language (UML) (see Clause 5.1.2) to present conceptual schemas for describing the spatial characteristics of geographic features.

This profile supports 1D and 2D. It satisfies the conformance classes A.1.1.1, A.1.1.2, A.1.1.3, A.2.1.1, A.2.1.2, A.3.1.1, A.3.1.2, A.4.1.1, and A.4.1.2. This profile conforms to level 1 of ISO 19106.

3 Normative references

The following normative documents contain provisions that, through reference in this text, constitute provisions of this profile.

ISO 19107 Geographic information — Spatial schema

ISO TS 19103 Geographic information — Conceptual schema language

ISO 19111 *Geographic information — Spatial referencing by coordinates*

4 Terms and definitions

Terms and definitions have been taken from ISO 19107. Only those which are specific to this profile have been included and modified where necessary to reflect the dimensional level used in this profile.

4.1

boundary
set that represents the limit of an entity

NOTE Boundary is most commonly used in the context of geometry, where the set is a collection of points or a collection of objects that represent those points.

4.2

closure

4.3 union of the interior and boundary of a topological or geometric object

4.4

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

4.5

composite curve

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

4.6

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.

4.7

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

4.8

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

4.9

connected node
node that starts or ends one or more edges

4.10

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.

4.11

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

4.12

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

4.13

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

4.14

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.

4.15

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.

4.16

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.

4.17

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

4.18

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.

4.19

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. Directed faces are used in the coboundary relation to maintain the spatial association between face and edge.

4.20

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".

4.21

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

4.22

domain
well-defined set

NOTE Domains are used to define the domain and range of operators and functions.

4.23

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.

4.24

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.

4.25

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

4.26

end point
last point of a curve

4.27

exterior

difference between the universe and the closure

NOTE The concept of exterior is applicable to both topological and geometric complexes.

4.28

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.

4.29

geometric aggregate
collection of geometric objects that has no internal structure

4.30

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

4.31

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 surface (2-D), the composition of the boundary operator in this definition terminates after at most two steps. It is also the case that the boundary of any object is a cycle.

4.31
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

4.32

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.

4.33

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, and surfaces.

4.34

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.

4.35

geometric set
set of direct positions

NOTE This set in most cases is infinite.

4.36

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.

4.37

isolated node
node not related to any edge

4.38

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

4.38
planar topological complex
topological complex that has a geometric realization that can be embedded in Euclidean 2 space.

4.39

point
0-dimensional geometric primitive, representing a position

NOTE The boundary of a point is the empty set.

4.40

ring
simple curve which is a cycle

NOTE Rings are used to describe boundary components of surfaces in 2-D coordinate systems.

4.41
set
unordered collection of related items (objects or values) with no repetition

4.41

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.

4.42

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

4.43

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

4.44 **4.45**

start point
first point of a curve

4.45

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.

4.46

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.

4.47

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

4.48

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.

4.49

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.

4.50

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

4.51

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.

4.52

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.

4.53

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

4.54

2.5 dimension
Two-dimensional topology used with a three-dimensional coordinate system constrained to a two-dimensional manifold

5 Symbols, notation and abbreviated terms

5.1 Presentation and notation

5.1.1 Unified Modelling Language (UML) concepts

In this Profile, conceptual schemas are presented in the Unified Modelling Language (UML).

For an overview of the use of UML in schema profiles, refer to ISO 19103 – Conceptual Schema Language.

5.1.2 Object Constraint Language (OCL)

The Object Constraint Language (OCL) is used to explicitly document all constraints, whether they are taken from ISO 19107 or are unique to this profile.

5.1.3 Naming Conventions

In this profile all packages and classes unique to this profile use the prefix FP (Functional Profile). All other classes making up the subset of ISO 19107 retain their GM and TP prefixes.

5.1.4 Textual Descriptions

In this profile the descriptions and explanations are generally translated from the more formal/theoretical text found in ISO 19107. This is in order to help the less experienced in this subject to more easily understand the content of this profile, for more formal, theoretically stringent descriptions please refer to ISO 19107.

5.2 Abbreviations

OCL Object Constraint Language

UML Unified Modelling Language

2-D Two-dimensional

2.5D Two and a half dimensional

ISO International Standards Organization

6 Geometry

6.1 FP Spatial Schema Packages

6.1.1 Semantics

This profile makes use of geometry packages defined in ISO 19107. Each package contains a number of geometry classes. The packages and their constituent classes are shown in Figure 3.

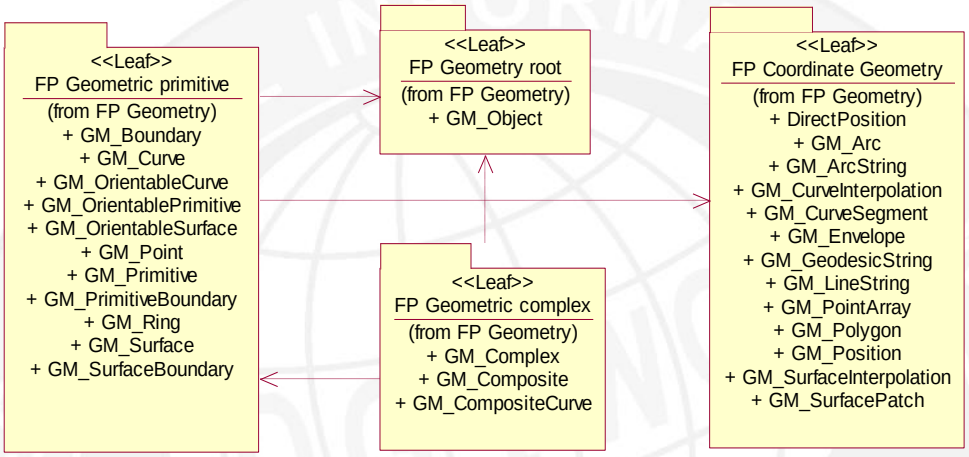


Figure 3 – FP Geometry Packages Overview

6.2 FP_Geometry Root Package

6.2.1 GM_Object (ISO 19107 Clause 6.2.2)

6.2.1.1 Semantics

GM_Object (Figure 4) is the abstract root class of all geometric object classes. GM_Objects are defined by a combination of a co-ordinate geometry (consisting of one or more of the geometry classes included in the FP_Geometry packages) and a SC_CRS (co-ordinate reference system). Instances (individual occurrences) of GM_Object are sets of direct positions (clause 6.6.2) in a particular SC_CRS.

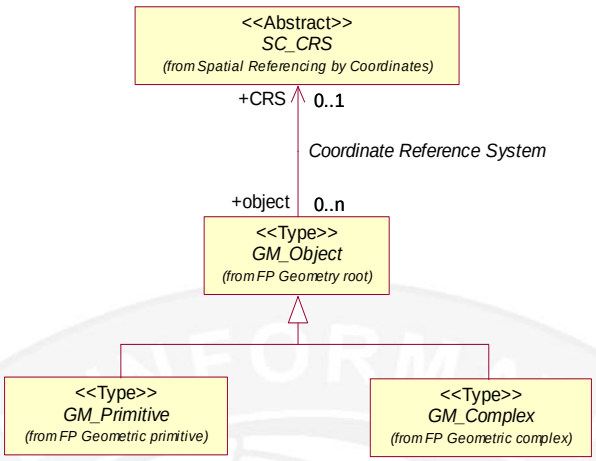


Figure 4 – GM_Object

6.2.1.2 Coordinate Reference System

The association "Coordinate Reference System" links this GM_Object to the co-ordinate reference system used by its DirectPosition co-ordinates. If this association is not used, then the GM_Object uses the SC_CRS from another GM_Object in which it is contained (of which it is a part). The SC_CRS is described more fully in the Spatial Referencing by Co-ordinates Profile.

```
GM_Object::Coordinate Reference System :: CRS[0..1]  : SC_CRS
```

6.3 FP Geometric primitive package

6.3.1 GM_Primitive (ISO 19107 Clause 6.3.10)

6.3.1.1 Semantics

GM_Primitive (Figure 5) is the abstract root class for all geometric primitives defined in this profile. A GM_Primitive is a GM_Object.

This profile uses six basic subtypes of GM_Primitive: GM_OrietablePrimitive GM_OrientableCurve, GM_OrientableSurface, GM_Point, GM_Curve and GM_Surface. There is no direct link between each GM_Primitive and the coordinate reference system SC_CRS used for defining the position of the GM_Primitive.

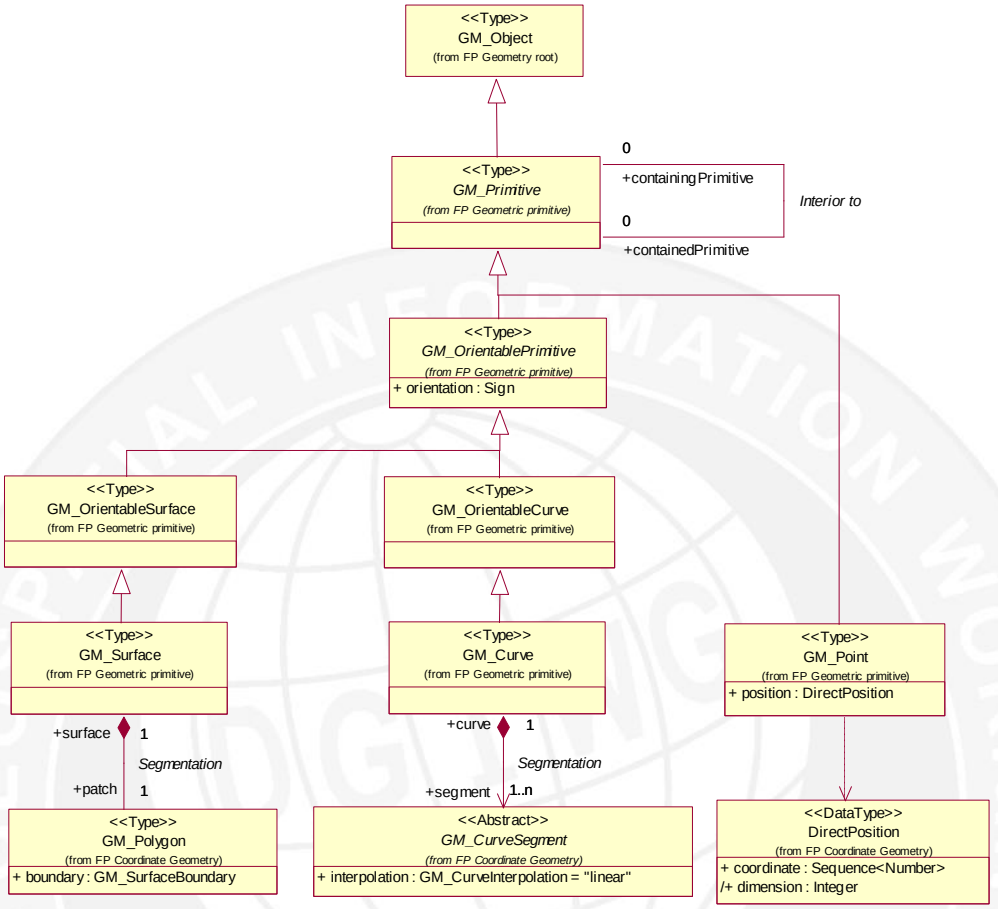


Figure 5 – Geometric primitives

6.3.1.2 Oriented

Each GM_Primitive of dimension 1 or 2 is associated to two GM_OrientablePrimitives, one for each possible orientation. 0-dimensional GM_Primitives have no associated GM_OrientablePrimitives. (see clause 6.5.3)

```
GM_Primitive::Oriented::proxy[0,2]:
Reference<GM_OrientablePrimitive>

alternate:  GM_Primitive::proxy[0,2] : GM_OrientablePrimitive
```

6.3.1.3 Interior to

The “Interior to” association has been included to satisfy conformance test A.1.1.3 of ISO19107 but the cardinality of both the “containedPrimitive” and “containingPrimitive” roles have been constrained to 0.

```
GM_Primitive::Interior to::containePrimitive[0]
```

```
GM_Primitive::Interior to::containingPrimitive[0]
```

6.3.1.4 Constraints

C1: There must not be a direct link between a GM_Primitive and the SC_CRS it uses.

```
context GM_Primitive inv :

self.AsType(GM_Object).CRS.isEmpty;
```

6.3.2 GM_Point (ISO 19107 Clause 6.3.11)

6.3.2.1 Semantics

GM_Point (Figure 5) is a 0-dimensional geometric primitive (GM_Primitive).

GM_Point is the data type for a geometric object consisting of one and only one point.

6.3.2.2 position

The “position” attribute defines the location of a GM_Point. The value of the attribute is the DataType *DirectPosition.*

```
GM_Point :: position : DirectPosition
```

6.3.2.3 Constraint

C2: 0-dimensional GM_Primitives have no associated GM_OrientablePrimitive.

```
context GM_Point inv :

self.AsType(GM_Primitive).proxy.isEmpty;
```

6.3.3 GM_OrientablePrimitive (ISO 19107 Clause 6.3.13)

6.3.3.1 Semantics

GM_OrientablePrimitive (Figure 5) is the abstract root class for all orientable geometric primitives. Orientable primitives are those that can have one of two orientations (either "+" or "-"), one being the mirror image of the other. There are two subtypes of orientable primitive: GM_OrientableCurve and GM_OrientableSurface. For curves, the orientation is the same as the direction in which the curve is traversed. For surfaces, when viewed from the "top" ("+" orientation) the exterior boundary appears counter clockwise.

6.3.3.2 orientation

The “orientation” attribute defines the direction of a GM_OrientablePrimitive and has a value “**+**” *or* “-“.

```
GM_OrientablePrimitive :: orientation : Sign
```

6.3.3.3 Oriented

The “Oriented” association links each GM_OrientablePrimitive with a GM_Primitive.

```
GM_OrientablePrimitive::Oriented::primitive[1]:Reference<GM_Primitiv
e>

alternative:

GM_Primitive::proxy[0,2] : Reference<GM_OrientablePrimitive>
```

6.3.4 GM_OrientableCurve (ISO 19107 Clause 6.3.14)

6.3.4.1 Semantics

A GM_OrientableCurve (Figure 5) is a one-dimensional GM_OrientablePrimitive that represents a specific orientation of a geometric curve.

6.3.4.2 orientation

The “orientation” attribute inherited from GM_OrientablePrimitive defines the direction of the curve which has the value “+” *or* “-“. A positive GM_OrientableCurve traverses its corresponding GM_Curve according to the digitization direction, a negative GM_OrientableCurve traverses its corresponding GM_Curve in the opposite direction. Where it forms the boundary of a surface, that surface is on the left of the GM_OrientableCurve.

6.3.4.3 Oriented

The “Oriented” association links each GM_ OrientableCurve with a GM_Curve.

6.3.4.4 Constraints

C3: The single GM_Primitive associated with a GM_OrientableCurve through the oriented association is a GM_Curve.

```
context GM_OrientableCurve inv :

self.AsType(GM_OrientablePrimitive).primitive.isKindOf(GM_Cur
ve);
```

C4: If the orientation is positive then the GM_OrientableCurve shall be its corresponding GM_Curve.

```
context GM_OrientableCurve inv :

self.orientation='+' implies

(       self.isTypeOf(GM_Curve)       and       self.asType
(GM_OrientableCurve).primitive=self.asType(GM_Curve) ) ;
```

6.3.5 GM_OrientableSurface (ISO 19107 Clause 6.3.15)

6.3.5.1 Semantics

A GM_ OrientableSurface (Figure 5) is a two-dimensional GM_OrientablePrimitive that represents a specific orientation of a geometric surface.

6.3.5.2 orientation

The "orientation" attribute inherited from GM_ OrientablePrimitive defines the direction of the surface which has the value + or -. A positive GM_OrientableSurface is the "top" side of its corresponding GM_Surface, a negative GM_ OrientableSurface is the "bottom" side of its corresponding GM_Surface.

6.3.5.3 Oriented

The "Oriented" association links each GM_ OrientableSurface with a GM_Surface.

6.3.5.4 Constraints

C5: The single GM_Primitive associated with a GM_OrientableSurface through the oriented association is a GM_Surface.

```
context GM_OrientableSurface inv :

self.AsType(GM_OrientableSurface).primitive.isKindOf(GM_Sur
face)
```

C6: If the orientation is positive then the GM_OrientableSurface shall be its corresponding GM_Surface.

```
context GM_ OrientableSurface inv :

self.orientation='+' implies

( self.isTypeOf(GM_Surface) and

self.asType(GM_OrientableSurface).primitive =

self.asType(GM_Surface) ) ;
```

6.3.6 GM_Curve (ISO 19107 Clause 6.3.16)

6.3.6.1 Semantics

A GM_Curve (Figure 5) is a 1-dimensional geometric primitive. It is a GM_OrientableCurve with a positive orientation.

A GM_Curve must be simple, i.e. it does not intersect itself.

6.3.6.2 Segmentation

The “Segmentation” association connects a GM_Curve to its GM_CurveSegments, which define its position, shape and orientation. It is a strong association, which means that an instance of GM_CurveSegment does not exist independently of the associated GM_Curve.

```
GM_Curve:: segment [1..n] : GM_CurveSegment
```

6.3.6.3 Constraint

C7: A GM_Curve does not intersect itself.

```
context GM_Curve inv :
self.AsType(GM_Object).isSimple[1]
```

6.3.7 GM_Surface(ISO 19107 Clause 6.3.17)

6.3.7.1 Semantics

GM_Surface (Figure 5) is a subclass of GM_Primitive and is the basis for 2-dimensional geometry. It is a GM_OrientableSurface with a positive orientation.

This profile is constrained to the use of a GM_Polygon as the surface patch for a GM_Surface.

6.3.7.2 Constraint

C8: A GM_Surface within this profile must only use a GM_Polygon as its GM_SurfacePatch.

[1] The isSimple operation is defined within ISO 19107 on GM_Object Type. The operation "isSimple" shall return TRUE if this GM_Object has no interior point of self-intersection or self tangency.

```
context GM_Surface inv :

self.isKindOf(GM_Polygon);
```

6.3.8 GM_Boundary (ISO 19107 Clause 6.3.2)

6.3.8.1 Semantics

The abstract class GM_Boundary (Figure 6) is the root data type for all the data types used to represent the boundary of geometric objects. Any subclass of GM_Object will use a subclass of GM_Boundary to represent its boundary through the operation `GM_Object::boundary`, which is changed to a composition association with a multiplicity of [0,1]. By the nature of geometry, boundary objects are cycles. A GM_Boundary (Figure 6) is a GM_Complex (clause 6.7.1).

6.3.8.2 Constraints

C9: A GM_Boundary is a cycle.

```
context GM_Boundary inv :

self.asType(GM_Object).isCycle[2]
```

6.3.9 GM_PrimitiveBoundary (ISO 19107 Clause 6.3.4)

6.3.9.1 Semantics

The abstract class GM_PrimitiveBoundary (Figure 6) is the root for the various types that describe the boundaries of subtypes of GM_Primitive. Each is a set of GM_Primitives containing all of the direct positions on the boundary of the GM_Primitive.

A GM_PrimitiveBoundary is a GM_Boundary.

6.3.10 GM_CurveBoundary (ISO 19107 Clause 6.3.5)

6.3.10.1 Semantics

The boundary of GM_Curves shall be represented as GM_CurveBoundary (Figure 6).

[2] The isCycle operation is defined within ISO 19107 on GM_Object Type. The operation "isCycle" shall return TRUE if this GM_Object has an empty boundary after topological simplification (removal of overlaps between components in non-structured aggregates, such as subclasses of GM_Aggregate).

6.3.10.2 startPoint, endpoint

A GM_CurveBoundary contains two GM_Point references.

```
GM_CurveBoundary::startPoint : Reference<GM_Point>;

GM_CurveBoundary::endPoint : Reference<GM_Point>;
```

6.3.11 GM_SurfaceBoundary (ISO 19107 Clause 6.3.7)

The boundary of GM_Surfaces shall be represented as GM_SurfaceBoundary (Figure 6).

A GM_SurfaceBoundary consists of references to a combination of one exterior GM_Ring and zero or more interior GM_Rings. The rings must be closed as described in clause 6.5.11.1.

6.3.11.1 exterior, interior

The roles “exterior” and “interior” of the associations linking a GM_Ring(s) to a GM_SurfaceBoundary defines the number of exterior and interior rings which form the boundary. These are strong aggregations which means that an instance of GM_Ring does not exist independently of the associated GM_SurfaceBoundary.

```
GM_SurfaceBoundary::exterior[1] : GM_Ring;

GM_SurfaceBoundary:: interior[0..n] : GM_Ring;
```

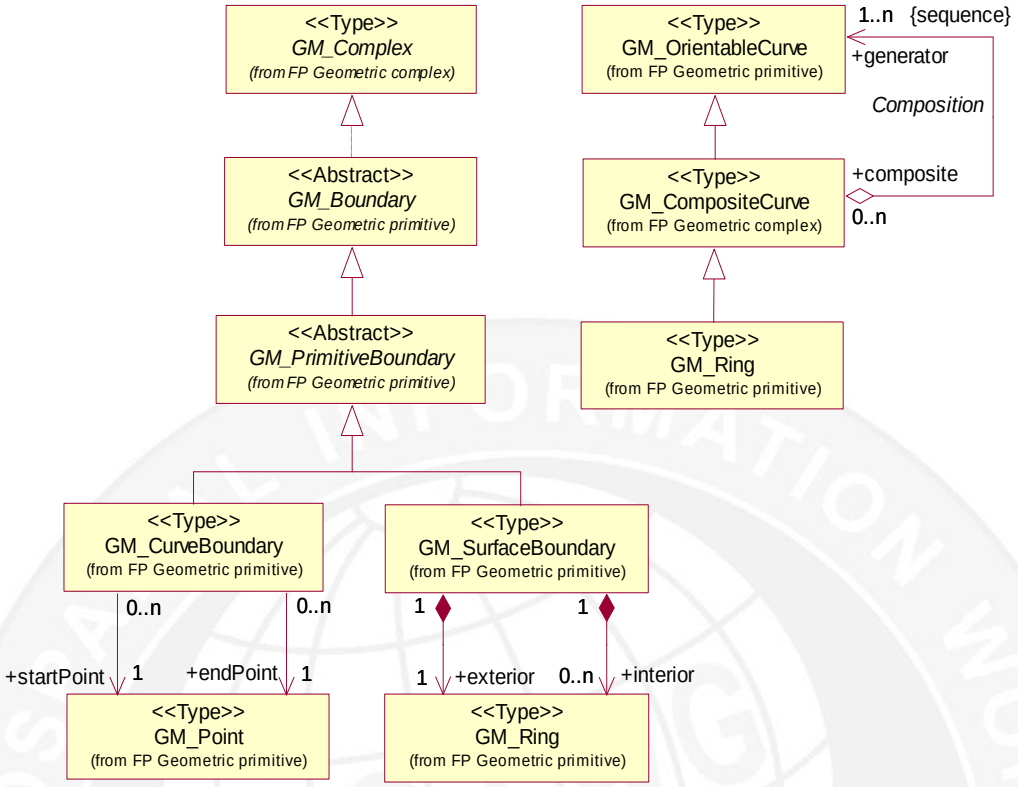


Figure 6 – Boundaries

6.3.12 GM_Ring (ISO 19107 Clause 6.3.6)

6.3.12.1 Semantics

A GM_Ring (Figure 6) is composed of a number of references to GM_OrientableCurves. The endPoint of GM_OrientableCurve “n” is the startPoint of GM_OrientableCurve “n+1” and the first startPoint is coincident with the last endPoint, i.e. the GM_Ring is closed. A GM_Ring must be simple, i.e. it does not intersect itself . Each GM_Ring must be oriented so that the associated GM_Surface is on the left.

6.3.12.2 Composition

The “Composition” association inherited from GM_CompositeCurve defines the GM_Ring as an ordrered set of references to GM_OrientableCurve.

```
GM_Ring::composition::generator[1..n]: GM_OrientableCurve
```

6.3.12.3 Constraints

C10: A GM_Ring is simple.

```
context GM_Ring inv :

self.AsType(GM_Object).isSimple
```

C11: A GM_Ring is either the exterior or interior part of a GM_SurfaceBoundary.

context GM_Ring inv :

```
(self->exists(sb: GM_SurfaceBoundary | sb.exterior=self)
and not

        self->exists(sb: GM_SurfaceBoundary |

                    sb.interior->intersection(self)=self))
or

(not       self->exists(sb:       GM_SurfaceBoundary       |
sb.exterior=self) and

        self->exists(sb: GM_SurfaceBoundary |

                    sb.interior->instersection(self)=self));
```

6.4 FP Coordinate Geometry Package

6.4.1 Coordinate Geometry and Coordinate Reference System

The classes within the Coordinate Geometry Package provide the different mechanisms necessary to define the position and shape of geometric primitives based on the use of coordinates. Coordinates values and other parameters such as interpolation methods, can only be interpreted with reference to a Coordinate Reference System.

ISO19111 provides a standardized model for the description of a Coordinate Reference System, SC_CRS, and ISO 19107 allows the identification of the SC_CRS to be used for interpreting the coordinate values and other parameters.

This profile is restricted to 2-D or 2.5-D coordinate systems where all instances of all geometric primitives are situated on the same underlying surface. This surface may be either an implicit underlying surface resulting from the definition of the coordinate reference system, or an explicit digital description of the surface of the earth such as a digital terrain model, or an implicit underlying surface derived from the union of all 0-dimensional and 1-dimensional primitives with the use of an interpolation method, usually a triangulation.

Explanation of the possible interpretation of the geometric construct is only provided, within this profile, for the following most common types of 2-D or 2.5-D coordinate reference system.

6.4.1.1 2-D Map or Grid coordinates

Each coordinate value is a pair of numbers, the first being the Easting, the second being the Northing, according to an identified map projection (e.g. UTM zone 31, Mercator). Description of such a coordinate reference system requires the identification of a geodetic datum and a map projection (CC_Conversion). Note that this profile does not allow the use of different map projections, such as different UTM zones, within the same collection of GM_Primitives.

The implicit underlying surface for this type of coordinate reference system will be the projection plane as a representation of the surface of the earth.

NOTE: The correct interpretation of the different geometric constructs will be very much dependent on the mathematical properties of the map projection (CC_Conversion). Application schemas with high positional accuracy requirements should note this fact.

6.4.1.2 2-D Geographic coordinates

Each coordinate value is a pair of numbers, the first being the Longitude, the second being the Latitude, referenced to an identified geodetic datum. Description of such a coordinate reference system requires the identification of a geodetic datum which relies on a specified ellipsoid.

The implicit underlying surface for this type of coordinate reference system will be the ellipsoid as a representation of the surface of the earth.

NOTE: The correct interpretation of the different geometric constructs will require correct use of geodetic computation. Application schemas with high positional accuracy requirements should note this fact.

6.4.1.3 2-D local coordinates

Each coordinate value is a pair of numbers, being the X and Y ordinates in a local cartesian reference system such as a digitization coordinate system.

The implicit underlying surface for this type of coordinate reference system will be the local X and Y plane which is not yet registered to the surface of the earth.

6.4.1.4 2-D Map or Grid coordinates with elevation or depth

Each coordinate value is a triplet of numbers, the first two numbers being 2-D Map or Grid coordinates as defined above, the last being an elevation or depth value according to a Vertical Datum . Description of such a coordinate reference system requires, beyond the identification of a geodetic datum and a map projection (CC_Conversion) required for 2-D Map Coordinates, the identification of the vertical datum. Note that this profile does not allow the use of different vertical datums within the same collection of GM_Primitives.

There is no implicit underlying surface for this type of coordinate reference system. A DTM (with its accompanying interpolation method) can be provided as an explicit underlying surface.

NOTE: The correct interpretation of the different geometric constructs will be very much dependent on the mathematical properties of the map projection (CC_Conversion), and when relevant on the DTM interpolation method. Application schemas with high positional accuracy requirements should note this fact.

6.4.1.5 2-D Geographic coordinates with elevation or depth

Each coordinate value is a triplet of numbers, the first two numbers being 2-D Geographic coordinates as defined above, the last being an elevation or depth value according to a Vertical Datum . Description of such a coordinate reference system requires, beyond the identification of a geodetic datum and associated ellipsoid as required for 2-D Geographic Coordinates, the identification of the vertical datum. Note that this profile does not allow the use of different vertical datums within the same collection of GM_Primitives.

There is no implicit underlying surface for this type of coordinate reference system. A DTM (with its accompanying interpolation method) can be provided as an explicit underlying surface.

NOTE: The correct interpretation of the different geometric constructs will require correct use of geodetic computation, and, when relevant of the DTM interpolation method. Application schemas with high positional accuracy requirements should note this fact.

6.4.1.6 3-D Geodetic coordinates

Each coordinate value is a triplet of numbers, the first two numbers being 2-D Geographic coordinates as defined above, the last being the ellipsoidal height. Description of such a coordinate reference system only requires the identification of a geodetic datum and associated ellipsoid as required for 2-D Geographic Coordinates.

There is no implicit underlying surface for this type of coordinate reference system. A DTM (with its accompanying interpolation method) can be provided as an explicit underlying surface.

NOTE: The correct interpretation of the different geometric constructs will require correct use of geodetic computation, and, when relevant of the DTM interpolation method. Application schemas with high positional accuracy requirements should pay attention to this fact.

6.4.1.7 3-D local coordinates

Each coordinate value is a triplet of numbers, being the X, Y and Z ordinates in a local cartesian reference system such as a digitization coordinate system.

The local cartesian coordinate system is not yet registered to the surface of the earth, nor to any vertical datum. There is no implicit underlying surface for this type of coordinate reference system.

6.4.2 Direct Position (ISO 19107 Clause 6.4.1)

6.4.2.1 Semantics

DirectPosition (Figure 7) holds the coordinates for a position within a particular coordinate reference system.

6.4.2.2 coordinate

The “coordinate” attribute defines the value of the coordinates in a coordinate reference system (e.g., 60.5, 0.7). The sequence length of the numbers and their value ranges must be in accordance with the definition of the specified coordinate reference system.

```
DirectPosition::coordinate : Sequence<Number>
```

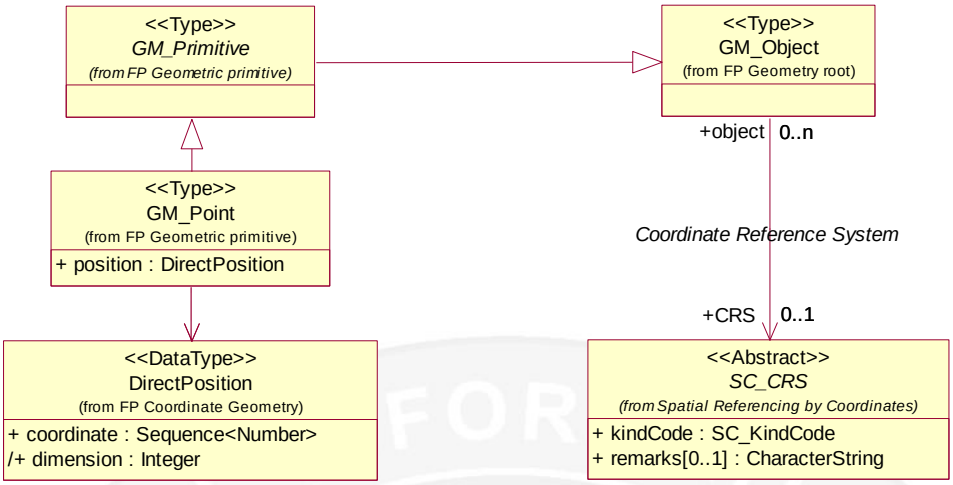


Figure 7 – Direct Position

6.4.2.3 dimension

The derived attribute “dimension”, is the length of the coordinate..

```
DirectPosition::dimension : Integer = (coordinate.length)
```

6.4.3 GM_Position (ISO 19107 Clause 6.4.5)

6.4.3.1 Semantics

The data type GM_Position (Figure 8) consists of either a DirectPosition or of a reference to a GM_Point (GM_PointRef) from which a DirectPosition can be obtained.

This profile prohibits the use of the indirect position (GM_PointRef).

```
GM_Position::direct [1] : DirectPosition
```

6.4.4 GM_PointArray (ISO 19107 Clause 6.4.6)

6.4.4.1 Semantics

An instance of the data type GM_PointArray (Figure 8) is used to contain a sequence of control points which are instances of DirectPosition.

```
GM_PointArray::column[1..n] : GM_Position
```

6.4.5 GM_Envelope (ISO 19107 Clause 6.4.3)

6.4.5.1 Semantics

GM_Envelope (Figure 8) is often referred to as a minimum bounding box or rectangle. Regardless of dimension (0-D, 1-D or 2-D), a GM_Envelope can be represented unambiguously as two direct positions (coordinate points). To encode a GM_Envelope, it is sufficient to encode these two points. This is consistent with all of the data types in this profile, their state is represented by their publicly accessible attributes.

6.4.5.2 upperCorner

The "upperCorner" of a GM_Envelope is a coordinate position consisting of all the maximal ordinates for each dimension for all points within the GM_Envelope.

```
GM_Envelope::upperCorner : DirectPosition
```

6.4.5.3 lowerCorner

The "lowerCorner" of a GM_Envelope is a coordinate position consisting of all the minimal ordinates for each dimension for all points within the GM_Envelope.

```
GM_Envelope::lowerCorner : DirectPosition
```

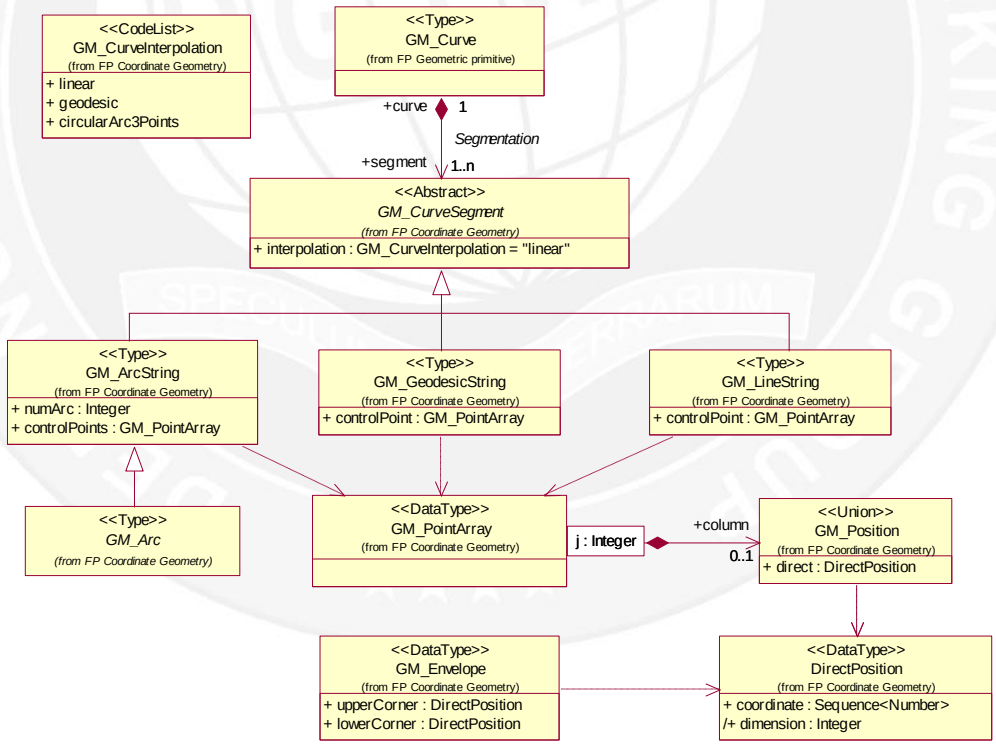


Figure 8 – Coordinate Geometry Classes

6.4.6 GM_CurveSegment (ISO 19107 Clause 6.4.9)

6.4.6.1 Semantics

A GM_CurveSegment (Figure 8) defines the position, shape and orientation of a single GM_Curve. A GM_CurveSegment consists either of positions which are joined by straight lines, or positions which fall on a line defined by a particular type of interpolation.

In this profile, four implementable subtypes of GM_CurveSegment can be used: GM_LineString, GM_GeodesicString, GM_ArcString and its subtype GM_Arc.

In this profile each GM_Curve is constrained to using a single GM_CurveSegment. Each GM_CurveSegment must lie on the underlying surface to which the GM_Curve belongs.

6.4.6.2 interpolation

The *interpolation* attribute defines the method of interpolation used by a GM_CurveSegment.

```
GM_CurveSegment::interpolation : GM_CurveInterpolation
```

6.4.6.3 Segmentation

The “Segmentation” association links a single GM_CurveSegment to its GM_Curve. It is a strong aggregation, which means that an instance of GM_CurveSegment does not exist independently of the associated GM_Curve.

```
GM_CurveSegment::Segmentation::curve [1] : GM_Curve
```

6.4.7 GM_CurveInterpolation (ISO 19107 Clause 6.4.8)

6.4.7.1 Semantics

GM_CurveInterpolation (Figure 8) is a list of codes, one of which is used to identify the interpolation mechanisms specified by an application schema.

In this profile, the types of "interpolation" available are constrained to the following:

a) Linear (linear) – the interpolation is defined by a series of DirectPositions on a straight line between each consecutive pair of controlPoints.

b) Geodesic (geodesic) – the interpolation mechanism shall return DirectPositions on a geodesic curve between each consecutive pair of controlPoints. A geodesic curve is a curve of shortest length. The geodesic shall be determined in the coordinate reference system of the GM_Curve in which the GM_CurveSegment is used.

c) Circular arc by 3 points (circularArc3Points) – the interpolation defined by a series of three DirectPositions on a circular arc passing from the start point through the middle point to the end point.for each set of three consecutive controlPoints. The middle point is located halfway between the start and end point.

6.4.8 GM_LineString (ISO 19107 Clause 6.4.10)

6.4.8.1 Semantics

GM_LineString (Figure 8) is a subtype of GM_CurveSegment which uses linear interpolation between each consecutive pair of control points controlPoint.

6.4.8.2 controlPoint

The “controlPoint” attribute defines the location of positions between which the curve is linearly interpolated. A GM_LineString can consist of a sequence of two or more control points which define its location. The order in which the control points occur defines the orientation of the GM_Curve which the GM_LineString references.

```
GM_LineString :: controlPoint : GM_PointArray
```

6.4.8.3 interpolation

The “interpolation” attribute is inherited from GM_CurveSegment. It is linear for GM_LineString.

6.4.8.4 Constraint

C12: A GM_LineString uses a "linear" interpolation method.

```
context GM_LineString inv :

self.asType(GM_CurveSegment).interpolation = 'linear';
```

6.4.9 GM_GeodesicString (ISO 19107 Clause 6.4.12)

6.4.9.1 Semantics

A GM_GeodesicString (Figure 8) is a subtype of GM_CurveSegment which uses geodesic as its method of interpolation.

6.4.9.2 controlPoint

The “controlPoint” attribute defines a sequence of two or more control points. The geodesic interpolation method is used between each pair of consecutive control points to define the location of the GM_GeodesicString. The order in which the control points occur defines the orientation of the related GM_Curve.

```
GM_GeodesicString :: controlPoint : GM_PointArray
```

6.4.9.3 interpolation

The “interpolation” attribute is inherited from GM_CurveSegment. It is geodesic for GM_GeodesicString.

6.4.9.4 Constraints

C13: A GM_GeodesicString uses a "geodesic" interpolation method.

```
context GM_GeodesicString inv :

self.asType(GM_CurveSegment).interpolation = 'geodesic';
```

6.4.10 GM_ArcString (ISO 19107 Clause 6.4.14)

6.4.10.1 Semantics

A GM_ArcString (Figure 8) is a subtype of GM_CurveSegment which uses circular arc interpolation between each consecutive triplet of control points controlPoint.

6.4.10.2 controlPoint

The “controlPoint” attribute defines a sequence of two or more control points. The circularArc3Points interpolation method is used between each triplet of consecutive control points to define the location the GM_ArcString. The order in which the control points occur defines the orientation of the GM_Curve which is composed of the GM_ArcString. *controlPoints* are treated as a sequence of overlapping sets of 3 GM_Positions, the start of each arc, some point between the start and end, and the end of each arc. Since the end of each arc is the start of the next, this GM_Position is not repeated in the controlPoint sequence.

```
GM_ArcString :: controlPoint : GM_PointArray
```

6.4.10.3 interpolation

The “interpolation” attribute is inherited from GM_CurveSegment. It is *circularArc3Points* for GM_ArcString.

6.4.10.4 numArc

The “numArc” attribute defines the number of circular arcs in the string. Since the interpolation method requires overlapping sets of 3 positions, the number of arcs determines the number of controlPoints.

```
GM_ArcString:numArc : Integer = ((controlPoint.length – 1)/2)
```

6.4.10.5 Constraints

C14: A GM_ArcString uses a "circularArc3Points" interpolation method.

```
context GM_ArcString inv :

self.asType(GM_CurveSegment).interpolation =
'circularArc3Points ';
```

6.4.11 GM_Arc (ISO 19107 Clause 6.4.15)

6.4.11.1 Semantics

A GM_Arc (Figure 8) is a subtype of GM_ArcString. It may be used as an alternative to a GM_ArcString if there is a requirement to return geometric information such as "center", "start angle", and "end angle".

6.4.12 GM_SurfacePatch (ISO 19107 Clause 6.4.34)

6.4.12.1 Semantics

The GM_SurfacePatch (Figure 9) is the abstract root class for all 2-dimensional geometric constructs. It uses a single interpolation method to define the shape and position of the associated GM_Surface primitives.

6.4.12.2 interpolation

The “interpolation” attribute defines the method of interpolation.

`GM_SurfacePatch::interpolation : GM_SurfaceInterpolation`

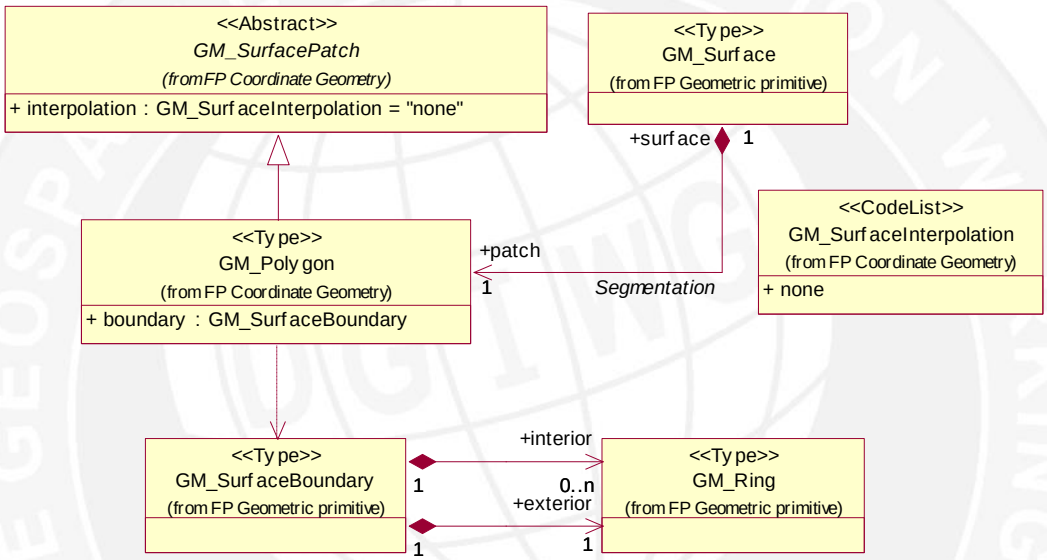


Figure 9 – GM_Polygon, GM_SurfaceInterpolation

6.4.13 GM_SurfaceInterpolation (ISO 19107 Clause 6.4.32)

6.4.13.1 Semantics

GM_SurfaceInterpolation (Figure 9) is a list of codes which are used to identify the method of interpolation.

In this profile, the types of interpolation are constrained to the following:

a) None (none) – the interior of the surface is not specified. The assumption is that the surface follows the reference surface defined by the coordinate reference system.

b) Planar (planar) – the interpolation is a section of a planar, or flat, surface. The boundary in this case shall be contained within that plane.

6.4.14 GM_Polygon (ISO 19107 Clause 6.4.36)

6.4.14.1 Semantics

A GM_Polygon (Figure 9) is defined by a boundary (see below) and an underlying surface to which this boundary is connected. The polygon uses planar interpolation. A GM_Polygon is a subtype of GM_SurfacePatch. It is the only type of surface patch used in this profile.

6.4.14.1 boundary

The “boundary” attribute defines the set of boundary curves as a GM_SurfaceBoundary, generally composed of one exterior GM_Ring and zero or more interior GM_Rings.

```
GM_Polygon::boundary : GM_SurfaceBoundary
```

6.4.14.2 Segmentation

A GM_Polygon defines the position and shape of a single GM_Surface. The association “Segmentation“ is a strong association, which means that an instance of GM_Polygon does not exist independently of its containing instance of GM_Surface. Although in ISO 19107 a GM_Surface can consist of one or more GM_Polygons, this profile constrains it to one.

```
GM_Polygon::Segmentation::surface [1] : GM_Surface
```

6.4.14.3 Constraint

C15: A GM_Surface is connected to a single GM_Polygon through the association Segmentation.

```
context GM_Surface inv :
self.patch->size = 1
```

6.4.15 GM_Complex (ISO 19107 Clause 6.6.2)

6.4.15.1 Semantics

A GM_Complex (Figure 10) is a collection of geometrically separate, simple GM_Primitives. If a GM_Primitive (other than a GM_Point) is in a particular GM_Complex, then there exists a set of primitives of lower dimension in the same complex that form the boundary of this primitive. For example a GM_Surface is a 2-dimentional object, its boundary consists of GM_Curves which are 1- dimensional.

6.4.15.2 Complex

The “Complex“ association defines the set of GM_Primitive elements composing the GM_Complex.

```
GM_Complex::Complex::element[1..n]: GM_Primitive
```

Within this profile, the element set of GM_Primitive is either a set of GM_Points, or a set of GM_Curves and GM_Points, or a set of GM_Surfaces, GM_Curves and GM_Points.

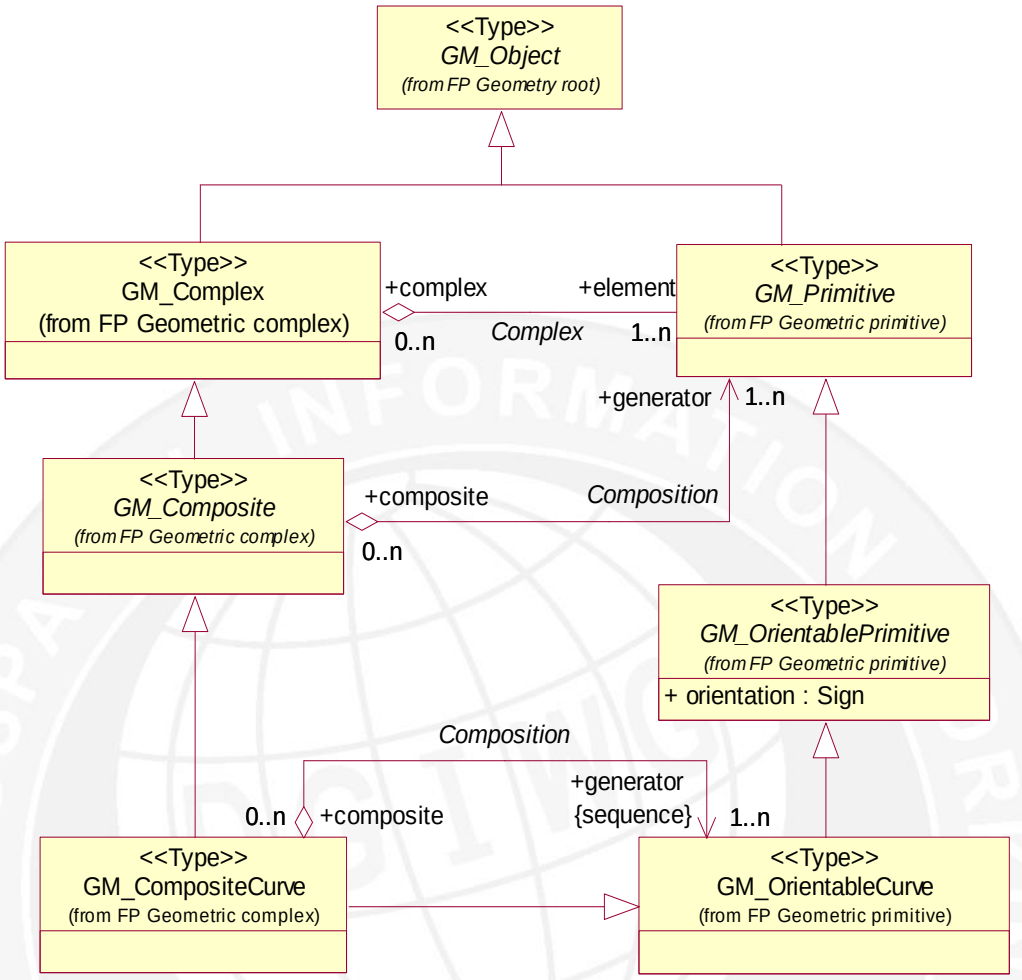


Figure 10 – GM_Complex

6.4.15.3 Constraints

C16: **If a GM_Surface belongs to a GM_Complex, the GM_Curves playing a role in its exterior and interior ring(s) also belong to the same GM_Complex.**

```
context GM_Complex inv :

self.AsType(GM_Complex).element

  ->forAll(a1: GM_Primitive |

  a1.IsKindOf(GM_Surface) implies

 a1.AsType(GM_Surface).patch.boundary.exterior
->union(a1.AsType(GM_Surface) ).patch.boundary.interior)

 ->forAll( r : GM_Ring | r.generator

                        ->forAll( ol : GM_OrientableCurve |

    self.AsType(GM_Complex).element-
>includesAll(ol.primitive))))
```

C17: **If a GM_Curve belongs to a GM_Complex, GM_Points with the same DirectPosition as its start point and end point also belong to the same GM_Complex.**

```
context GM_Complex inv :

self.AsType(GM_Complex).element

  ->forAll(l1: GM_Primitive |

  l1.IsKindOf(GM_Curve) implies

  self.AsType(GM_Complex).element->exists(p : GM_Point |

  p.position                                                    =
l1.AsType(GM_Curve).boundary.startpoint.position )

  and

  self.AsType(GM_Complex).element->exists(p : GM_Point |

  p.position  =  l1.AsType(GM_Curve).boundary.endpoint.position
))
```

C18: **Two different GM_Surfaces belonging to the same GM_Complex are not coincident, do not overlap each other, nor are included in one another.**

```
context GM_Complex inv :

self.AsType(GM_Complex).element

  ->forAll(a1,a2 : GM_Primitive |

  a1.IsKindOf(GM_Surface)                                    and
                a2.IsKindOf(GM_ Surface) implies

   not
a1.AsType(GM_Object).intersects(a2.AsType(GM_Object))

   and                                                       not
a1.AsType(GM_Object).equals(a2.AsType(GM_Object)))
```

C19: **Two different GM_Curves belonging to the same GM_Complex are not coincident, do not cross nor are tangent to each other. They can only be connected at their start or end point.**

```
context GM_Complex inv :

self.AsType(GM_Complex).element

  ->forAll(l1,l2 : GM_Primitive |

  l1.IsKindOf(GM_Curve) and l2.IsKindOf(GM_Curve) implies

  not  l1.AsType(GM_Object).intersects(l2.AsType(GM_Object))
)
```

C20: **Two different GM_Points belonging to the same GM_Complex may not be coincident**

```
context GM_Complex inv :

self.AsType(GM_Complex).element

  ->forAll(p1,p2 : GM_Primitive |

  p1.IsKindOf(GM_Point)  and  p2.IsKindOf(GM_Point)  implies
       p1.AsType(GM_Point).position                          <>
p2.AsType(GM_Point).position)
```

C21: **If a GM_Curve belongs to the same GM_Complex as a GM_Surface, it does not cross the boundary of this GM_Surface. It may either be totally disjoint or be part of the boundary of this GM_Surface.**

```
context GM_Complex inv :
self.AsType(GM_Complex).element
   ->forAll(l1, a2: GM_Primitive |
   l1.IsKindOf(GM_Curve) and a2.IsKindOf(GM_Surface)
   implies
   not  l1.AsType(GM_Object).intersects(a2.AsType(GM_Object))
)
```

NOTE If the GM_Curve is totally interior to the GM_ Surface, then it is part of an interior GM_Ring, eventually reduced to the two GM_OrientableCurves based on this single GM_Curve.

C22: **If a GM_Point belongs to the same GM_Complex as a GM_Curve, it may not be coincident with any interior point of this GM_Curve. It may either be totally disjoint or be coincident with this GM_Curve start or end point.**

```
context GM_Complex inv :
self.AsType(GM_Complex).element
   ->forAll(p1, l2: GM_Primitive |
   p1.IsKindOf(GM_Point) and l2.IsKindOf(GM_Curve) implies
   not  p1.AsType(GM_Object).intersects(l2.AsType(GM_Object))
)
```

6.4.16 GM_Composite (ISO 19107 Clause 6.6.3)

6.4.16.1 Semantics

A geometric composite, GM_Composite (Figure 10), is a collection of primitives which is logically equivalent to a single primitive and which could exist as a single example of that primitive. For example, a composite curve is a collection of curves which could be equally represented by a single curve. This does not apply to GM_Point which can contain one and only one point.

6.4.17 GM_CompositeCurve (ISO 19107 Clause 6.6.5)

6.4.17.1 Semantics

A GM_CompositeCurve (Figure 10) has all the geometric properties of a curve. A composite curve is a sequence of GM_OrientableCurves (using the same orientation), each curve (except the first) begins where the previous curve ends.

6.4.17.2 generator

The “Composition::generator“ association role is a collection of GM_OrientableCurves which, when joined, form the core geometry of this composite.

```
GM_CompositeCurve::composition:::generator                    [1.*]:
Sequence<GM_OrientableCurve>
```

7 Topology

7.1 FP Topological Primitive Package

This profile makes use of topology packages defined in ISO 19107. Each package contains a number of topology classes. The packages and their constituent classes are shown in Figure 11.

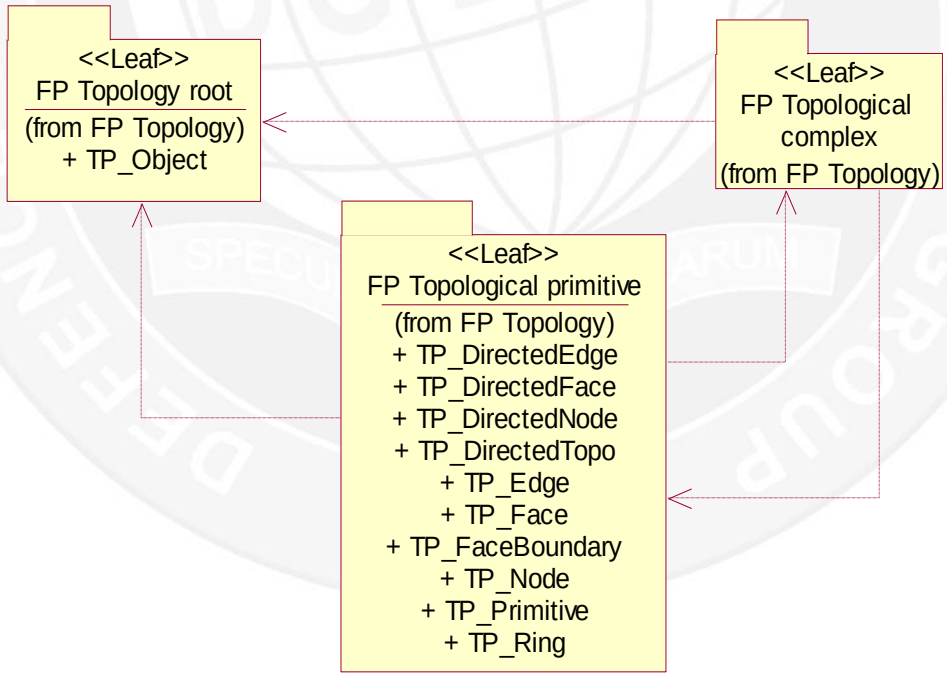


Figure 11 – FP Topology Packages Overview

7.1.1 TP_Object (ISO 19107 Clause 7.2.2)

7.1.1.1 Semantics

Topological object, TP_Object (Figure 12) is an abstract class that supplies a root type for topological complexes and topological primitives.

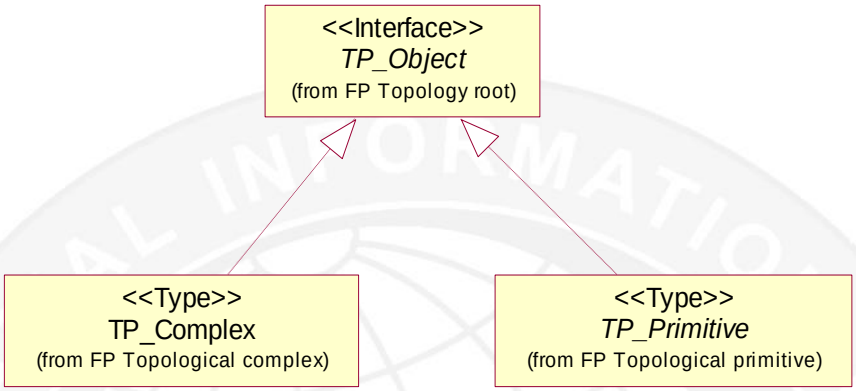


Figure 12 – TP_Object

7.1.2 TP_Primitive (ISO 19107 Clause 7.3.10)

7.1.2.1 Semantics

TP_Primitive (Figure 13) is the abstract root class for all topological primitives. This profile uses three subtypes of TP_Primitive, TP_Node (0-dimensional), TP_Edge (1-dimensional) and TP_Surface (2-dimensional). Within this profile, each topological primitive has a one-to-one relationship, known as a “a direct geometric realisation” , with a corresponding geometric primitive.

This profile considers all topological primitives to be part of at least one topological complex. The complex and its primitives must have a one to one correspondence with a geometric complex and its equivalent geometric primitives.

```
TP_Primitive::geometry[1] : GM_Primitive

TP_Complex::geometry[1] : GM_Complex
```

Geometric and topological complexes are discussed further in clauses 6.7.1 and 7.9.1 respectively.

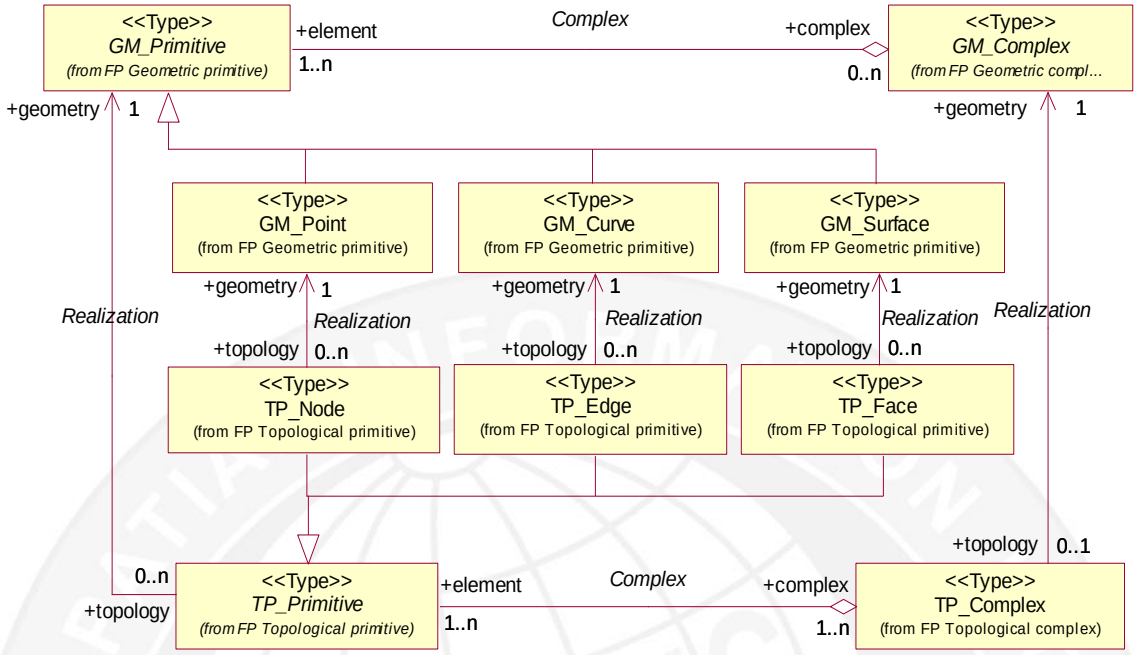


Figure 13 – Topological primitives and geometric realizations

7.1.2.2 Realization

The “Realization“ association defines the GM_Primitives which are the geometric realizations of their respective TP_Primitives.

```
TP_Primitive:: Geometry [1] : GM_Primitive

GM_Primitive [0..n] : TP_Primitive
```

7.1.2.3 Complex

The “Complex“ association defines the TP_Complex(s) to which a TP_Primitive belongs.

A TP_Primitive may belong to one or more TP_Complex(s).

```
TP_Primitive::Complex::complex [1..*] : Reference<TP_Complex>
```

7.1.2.4 Constraint

C23: **A TP_Primitive (TP_Node, TP_Edge or TP_Face) will have geometric realization that is a 1-to-1 relationship to its associated GM_Primitive (GM_Point, GM_Curve, GM_Surface).**

```
context TP_Primitive inv :

self.AsType(TP_Geometry).geometry                                    =
self.AsType(GM_Primitive);
```

7.1.3 TP_DirectedTopo (ISO 19107 Clause 7.3.11)

7.1.3.1 Semantics

TP_DirectedTopo (Figure 14) is the abstract root class for all directed topological primitives.

From a computational point of view, elements of TP_DirectedTopo are equivalent to the various orientable geometric objects (GM_OrientableObject) in the geometry packages, i.e. GM_OrientableCurve and GM_OrientableSurface. TP_DirectedNode does not have a geometric object equivalent.

As is the case with geometry, each topological primitive inherits its orientation from its corresponding directed topological primitive. This means that TP_Node is equivalent to a positive TP_DirectedNode, a TP_Edge to a positive TP_DirectedEdge, etc.

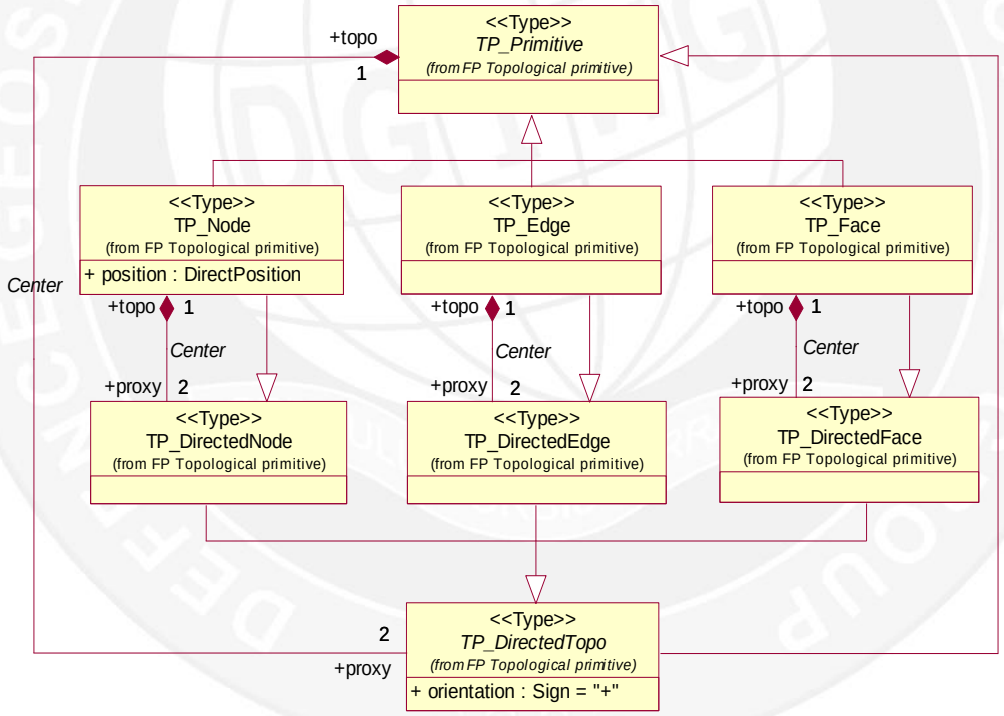


Figure 14 – TP_DirectedTopo subclasses

7.1.3.2 Center

The "Center" association links each TP_Primitive with its two corresponding TP_DirectedTopo.

```
TP_Primitive::center::proxy [2] : TP_DirectedTopo.
```

7.1.3.3 orientation

The orientation of a TP_DirectedNode, TP_DirectedEdge or TP_DirectedFace is provided by the orientation attribute inherited from its TP_Directed Topo parent.

```
TP_DirectedTopo::orientation : sign = "+"
```

7.1.3.4 Constraint

C24: A TP_Primitive is a TP_DirectedTopo with a positive orientation.

```
context TP_Primitive inv :

self.AsType(TP_DirectedTopo).orientation = '+';
```

C25: A TP_Primitive is connected to a positive and a negative TP_DirectedTopo through the center association

```
context TP_Primitive inv :

self.proxy->select(orientation = '+')= 1 and
        self.proxy->select(orientation = '-')= 1
```

7.1.4 TP_Node (ISO 19107 Clause 7.3.12)

7.1.4.1 Semantics

TP_Node (Figure 15) is a 0-dimensional TP_Primitive. A TP_Node has a geometric realization that is a GM_Point. A TP_Node is a TP_DirectedNode with a positive orientation.

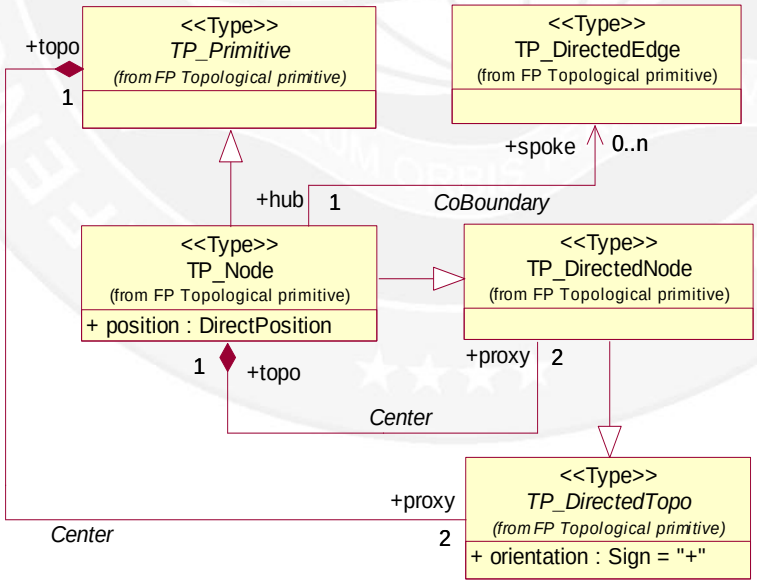


Figure 15 – TP_Node

7.1.4.2 position

The “position“ attribute provides the location of a TP_Node through the realization of its GM_Point equivilent.

```
GM_Point :: position : DirectPosition
```

7.1.4.3 Center

The “Center“ association relates each TP_Node *to* its two corresponding TP_DirectedNodes.

```
TP_Node::center :: proxy[2]  : TP_DirectedNode
```

7.1.4.4 CoBoundary

The “CoBoundary“ association connects each TP_Node with its positive or negative TP_DirectedEdge which indicate the two possible orientations of a TP_Edge. A positive TP_DirectedEdge enters a node and a negative TP_DirectedEdge leaves a node (see Figure 14).

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

     (coboundary::hub[1] : TP_Face
```

If the TP_Node has no coboundary relationship to TP_DirectedEdges, then the TP_Node is an isolated node.

```
TP_Node::isolatedIn ::container[0..1]:TP_Face
```

7.1.4.5 Constraints

C26: An isolated TP_Node has an empty coboundary.

```
context TP_Node inv :

self.spoke->size=0 or

self.AsType(TP_Primitive).container->size=0;
```

7.1.5 TP_DirectedNode (ISO 19107 Clause 7.3.13)

7.1.5.1 Semantics

A TP_DirectedNode (Figure 15) is connected to a TP_Node through the “center“ association. TP_DirectedNode is a supertype of TP_Node.

7.1.5.2 Center

The “Center“ association relates each TP_DirectedNode to its corresponding primitive TP_Node.

```
TP_DirectedNode::center :: topo[1]  : TP_Node
```

7.1.6 TP_Edge (ISO 19107 Clause 7.3.14)

7.1.6.1 Semantics

TP_Edge (Figure 17) is a 1-dimensional TP_Primitive. *A* TP_Edge has a geometric realization that is a GM_Curve. The position, shape and orientation of a TP_Edge is defined by the single GM_CurveSegment / GM_Curve relationship of its geometric realization. The boundary of a TP_Edge consists of a pair of nodes, one at the start of the edge (negative TP_DirectedNode) and one at the end (positive TP_DirectedNode) see Figure 16.

7.1.6.2 Center

The “Center“ association relates each TP_Edge to its two corresponding TP_DirectedEdges

```
TP_Edge :: center :: proxy[2]  : TP_DirectedEdge
```

7.1.6.3 Boundary

The “boundary“. association connects each TP_Edge with its start (negative) and end (positive)TP_DirectedNode. Orientation of the TP_DirectedNode must be in accordance with its start (negative) or end (positive) position see (Figure 17).

```
TP_Edge:: boundary :: boundary[2]: TP_DirectedNode

     (TP_DirectedNode :: boundary  :: primitive [0..n]  : TP_Edge)
```

7.1.6.4 coBoundary

The “coBoundary“ association defines whether a TP_Face is on the left (positive TP_DirectedFace) or on the right (negative TP_DirectedFace) of each TP_Edge.

```
TP_Edge :: coboundary :: spoke[0,2]  : TP_DirectedFace

     (coboundary :: hub[1..n]  : TP_Edge)
```

7.1.6.5 Constraint

C27: The geometric realization of the start and end TP_Nodes of a TP_Edge are the start and end control points of the GM_Curve.

```
context TP_Edge inv :

self.boundary->forAll(dn :TP_DirectedNode|

   (dn.asType(TP_DirectedTopo).orientation = ”-” and

   dn.topo.position
=self.AsType(GM_Curve).segment.controlPoint[1])

or(dn.orientation =”+” and

   dn.topo.position = self.AsType(GM_Curve).segment

.controlPoint[self.AsType(GM_Curve).segment.controlPoint-
>size])))
```

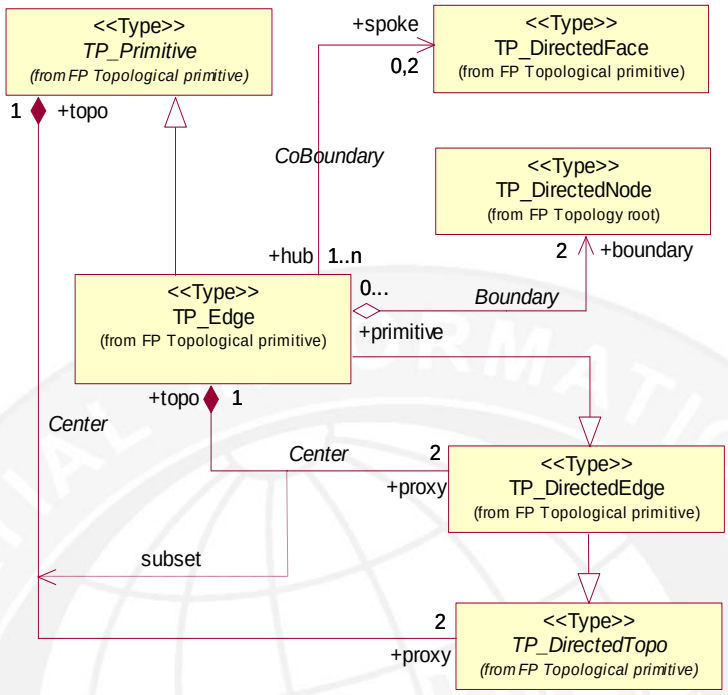


Figure 16 – TP_Edge

7.1.7 TP_DirectedEdge (ISO 19107 Clause 7.3.15)

7.1.7.1 Semantics

A TP_DirectedEdge (Figure 17) represents an association between a TP_Edge and one of its orientations.

7.1.7.2 Center

The “center“ association relates each TP_DirectedEdge to its corresponding TP_Edge.

```
TP_DirectedEdge::center :: topo[1] : TP_Edge
```

7.1.8 TP_Face (ISO 19107 Clause 7.3.16)

7.1.8.1 Semantics

TP_Face (Figure 18) is a 2-dimensional TP_Primitive. Its geometric realization is a GM_Polygon..

7.1.8.2 Center

The “Center“ association relates each TP_Face to its two corresponding TP_DirectedFace.

```
TP_Face:: proxy[2] : TP_DirectedFace
```

```
alternate:  TP_DirectedFace::topo[1] : Reference<TP_Face>
```

7.1.8.3 Boundary

The “boundary“ association connects each TP_Face with its bounding TP_DirectedEdge. The orientation of a TP_DirectedEdge must be positive if the face lies on the left of the TP_Edge, negative if it lies on the right.

```
TP_Face::boundary :: boundary[1..*]  : TP_DirectedEdge
```

7.1.8.4 Constraints

C28: The composition of exterior and interior GM_Ring(s) bounding the geometric realization of a TP_Face is based on the geometric realization of its bounding TP_Edges .

```
context TP_Face inv :

self.AsType(GM_Surface).patch.boundary.exterior

->union(self.AsType(GM_Surface).patch.boundary.interior)
  -> forAll(r : GM_Ring |

        r.asType(GM_CompositeCurve).generator

        ->forAll(ol : GM_OrientableCurve |

            self.boundary

            ->exists ( de:TP_DirectedEdge |
        de.topo.asType(GM_Curve)=ol.primitive ) )  )

and

self.boundary

  ->forall ( de:TP_DirectedEdge |

        self.AsType(GM_Surface).patch.boundary.exterior

  ->union(self.AsType(GM_Surface).patch.boundary.interior)
        ->exists(r : GM_Ring |

r.asType(GM_CompositeCurve).generator

->includes(de.topo.asType(GM_OrientableCurve))

        or

r.asType(GM_CompositeCurve).generator

->includes(de.topo.asType(GM_OrientablePrimitive)
        .negate.asType(GM_OrientableCurve)) ) ) )
```

C29: A TP_Face is not isolated in any other TP_Primitive

```
context TP_Face inv :

self.AsType(TP_Primitive).container->size=0
```

C30: **A TP_Edge is not isolated in any other TP_Primitive (for topologies up to 2D)**

```
context TP_Edge inv :

self.AsType(TP_Primitive).container->size=0
```

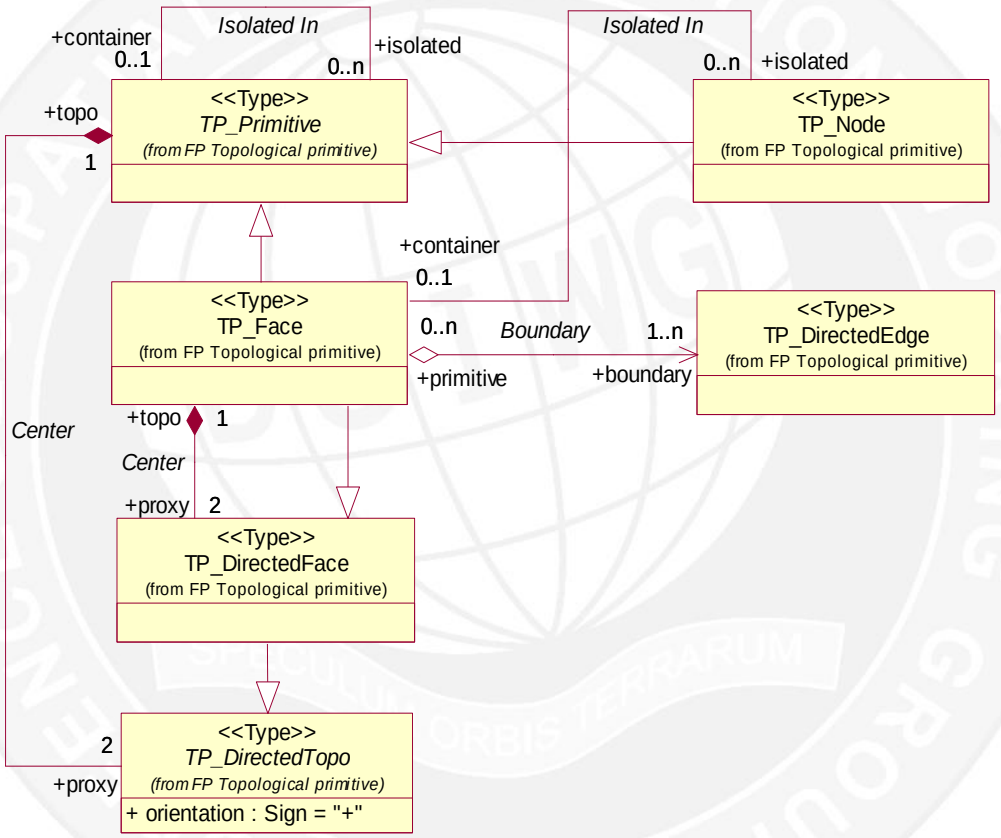


Figure 17 – TP_Face

7.1.9 TP_DirectedFace (ISO 19107 clause 7.3.17)

7.1.9.1 Semantics

A TP_DirectedFace is a TP_DirectedTopo connected to a TP_Face through the center association. TP_DirectedFace is a supertype of TP_Face

7.1.9.2 Center

The “Center“ association relates each TP_DirectedFace to its corresponding primitive TP_Face.

```
TP_DirectedFace:: center :: topo[1]  : TP_Face
```

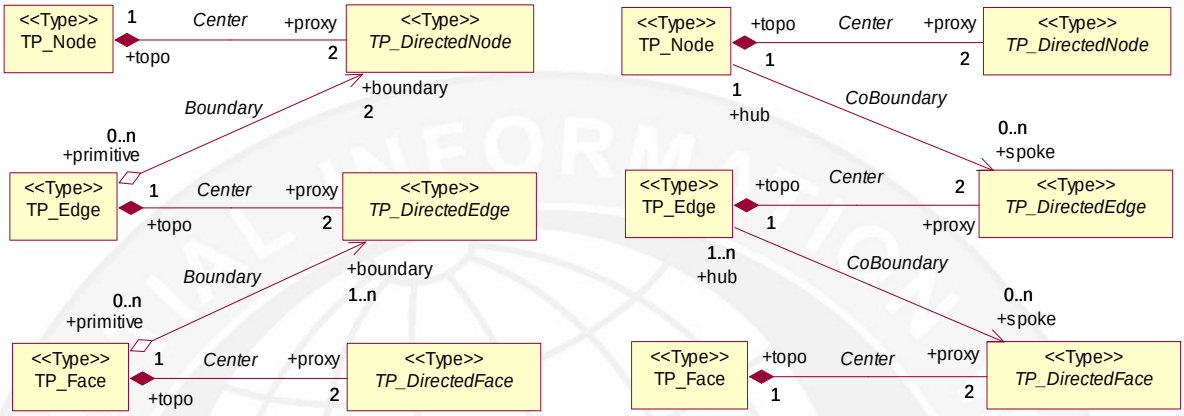


Figure 18 – Boundary & CoBoundary Associations

7.1.10 TP_Boundary (ISO 19107 clause 7.3.2)

7.1.10.1 Semantics

The class TP_Boundary (Figure 19) is the root data type for all the data types used to represent the boundary of topological objects. Any subclass of TP_Object will use a subclass of TP_Boundary to represent its boundary through the operation `TP_Object::boundary`, which is changed to a composition association with a multiplicity of [0,1]. By the nature of topology, boundary objects are cycles.

7.1.10.2 Constraints

C31: A TP_Boundary is a cycle.

```
context TP_Boundary inv :

self.asType(TP_Expression).isCycle
```

7.1.11 TP_PrimitiveBoundary (ISO 19107 clause 7.3.4)

7.1.11.1 Semantics

The abstract class TP_PrimitiveBoundary (Figure 19) is the root for the various types that describe the boundaries of subtypes of TP_Primitive. Each is a cyclic set of TP_Primitives of dimension n-1 that bound an n dimensional primitive.

A TP_PrimitiveBoundary is a TP_Boundary.

7.1.12 TP_EdgeBoundary (ISO 19107 clause 7.3.5)

7.1.12.1 Semantics

The boundary of TP_Edges shall be represented as TP_EdgeBoundary (Figure 19).

7.1.12.2 startNode, endNode

A TP_EdgeBoundary contains two TP_DirectedNodes.

```
TP_EdgeBoundary::startNode : TP_DirectedNode;
```

```
TP_EdgeBoundary::endNode : TP_DirectedNode;
```

7.1.13 TP_FaceBoundary (ISO 19107 clause 7.3.6)

7.1.13.1 Semantics

A TP_FaceBoundary (Figure 19) consists of one or more TP_Rings, forming the elements of its boundary. Normally one ring will be exterior, this may not always be possible in a general manifold in which case all boundaries must be interior.

Each ring is oriented such that the face is on its left.

7.1.13.2 exterior, interior

The roles “exterior” and “interior” of the associations linking a TP_Ring(s) to a TP_FaceBoundary defines the number of exterior and interior rings which form the boundary.

```
TP_FaceBoundary::exterior[0,1] : TP_Ring;
```

```
TP_FaceBoundary::interior[0..*] : TP_Ring;
```

7.1.14 TP_Ring (ISO 19107 clause 7.3.8)

7.1.14.1 Semantics

A TP_Ring (Figure 19) represents a single element of a TP_FaceBoundary. It consists of a number of TP_DirectedEdges connected in a cycle.

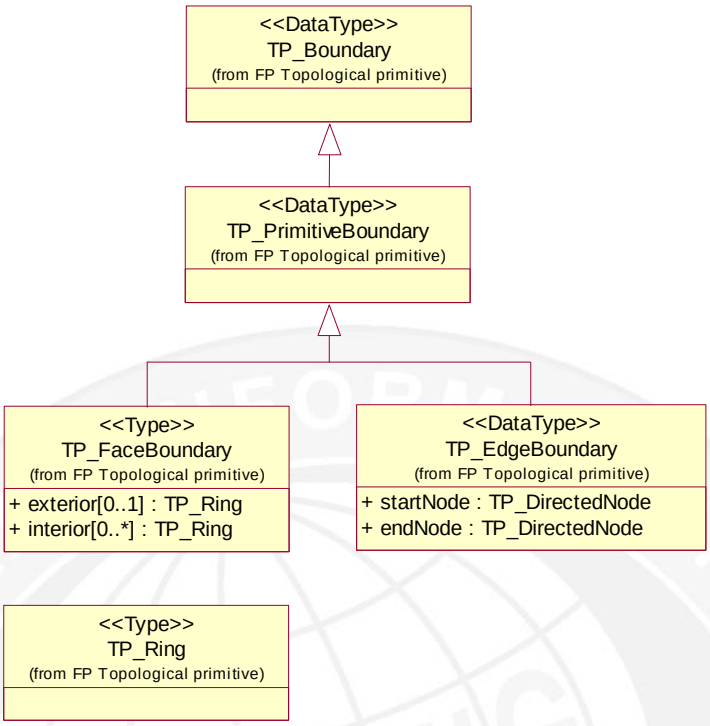


Figure 19 – TP_Boundary

7.2 TP Complex Package

7.2.1 TP_Complex (ISO 19107 clause 7.4.2)

7.2.1.1 Semantics

Topological complexes parallel the geometric complexes introduced in clause 6.7.1. A TP_Complex (Figure 13) may use set operations on its elements to perform the equivalent set operations on the underlying sets of direct positions that are represented by the geometric elements of a geometric realization (a GM_Complex).

7.2.1.2 Complex

The “Complex“ association defines the set of TP_Primitive elements composing the TP_Complex.

```
TP_Complex::Complex::element[1..*]: TP_Primitive
```

Within this profile, the element set of TP_Primitive is either a set of TP_Nodes, or a set of TP_Edges and TP_Nodes, or a set of TP_Faces and TP_Edges and TP_Nodes.

The one-way Realization association defines the GM_Complex which is the geometric realization of each TP_Complex

```
TP_Complex::Realization::geometry[1]: GM_Complex

    GM_Complex::Realization::topology[0..1]: TP_Complex
```

7.2.1.3 Constraints

C32: **If a TP_Edge belongs to a TPComplex, its start and end TP_Nodes belong to the same TP_Complex.**

```
Context TP_Complex inv:

self.element

  ->forAll(e1: TP_Primitive |

        e1.IsKindOf(TP_Edge) implies

              self.AsType(TP_Complex).element

              -
>includes(e1.AsType(TP_Edge).boundary[1].topo)

        and

              self.AsType(TP_Complex).element

              -
>includes(e1.AsType(TP_Edge).boundary[2].topo))
```

C33: **If a TP_Face belongs to a TP_Complex, its bounding TP_Edges belong to the same TP_Complex.**

```
self.element

  ->forAll(f1: TP_Primitive |

  f1.IsKindOf(TP_Face) implies

  f1.AsType(TP_Face).boundary->forAll(de  :  TP_DirectedEdge
|

        self.AsType(TP_Complex).element->select(e2:
TP_Primitive |

        e2.IsKindOf(TP_Edge))->includes(de.topo) ) )
```

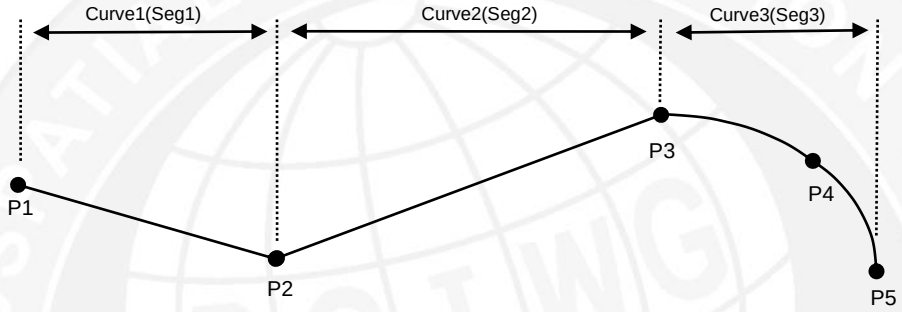
Annex A

(informative)

Examples

A.1 Geometry

A.1.1 Curve Example



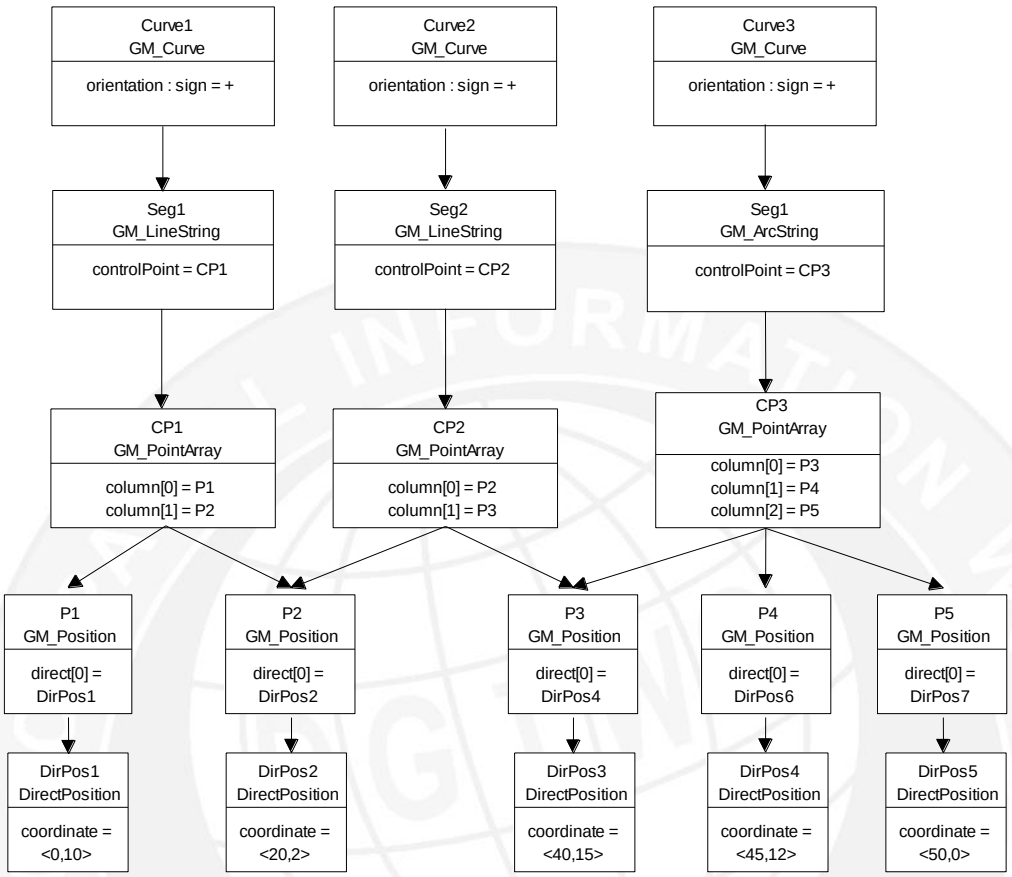


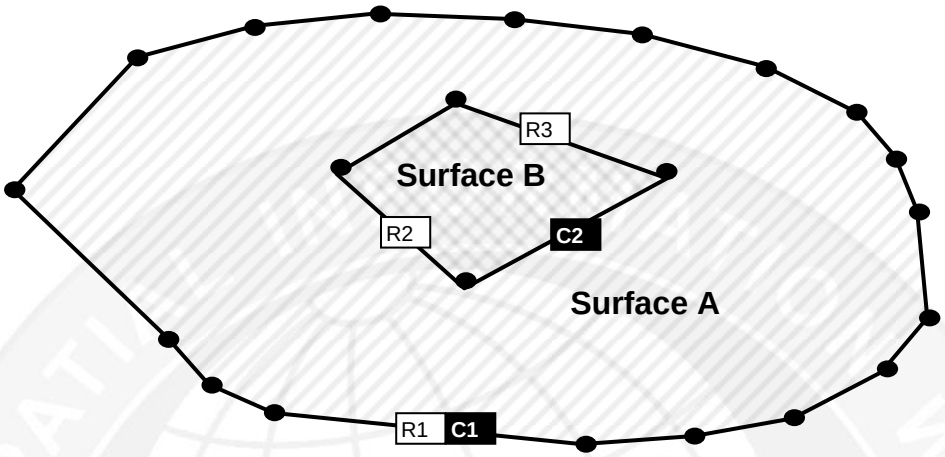
Figure A.1 – Curve Example

The following describes the geometrical elements of the curve example (Figure A.1).

The GM_Curves 1 and 2 are made up of GM_CurveSegments which holds their X,Y positions in GM_LineStrings as GM_PointArrays. GM_Curve 3 is an arc whose GM_CurveSegment holds its X,Y positions in a GM_ArcString as a GM_PointArray.

The interpolation used by the segment is provided by the “code list” GM_CurveInterpolation. Figure A.1 depicts both “linear” and “circular arc” interpolation. The direction of the curve is defined by the direction of digitizing. A GM_Curve may consist of more than one GM_CurveSegment (e.g. a mixture of linear and arc interpolation) but is constrained to one per GM_Curve in this profile. Each GM_Curve has a start and end point which are the first and last control points of the GM_LineString. If the curve is cyclic (closed) then the start and end point are the same.

A.1.2 Surface Example



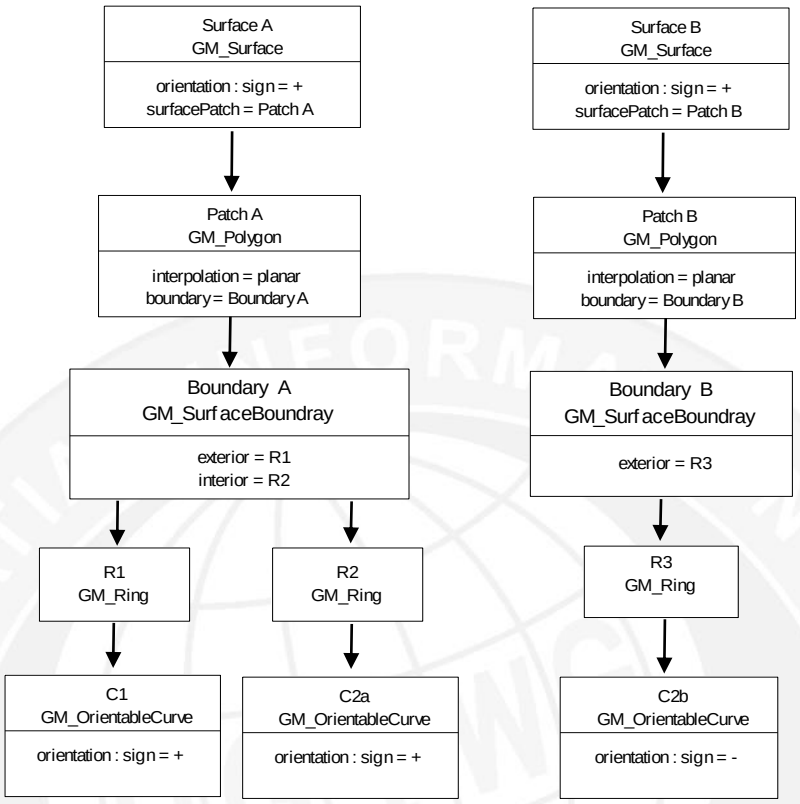


Figure A.2 – Surface Example

The following describes the geometrical elements of the surface example (Figure A.2).

Surface A is represented by the surface patch GM_Polygon (Patch A). Its GM_Surface boundary (Boundary A) is comprised of two rings; exterior ring R1 and interior ring R2. The rings in turn are comprised of GM_Curves.

Curve C1 forms the absolute external boundary and R1 consists of a GM_Curve with a positive orientation.

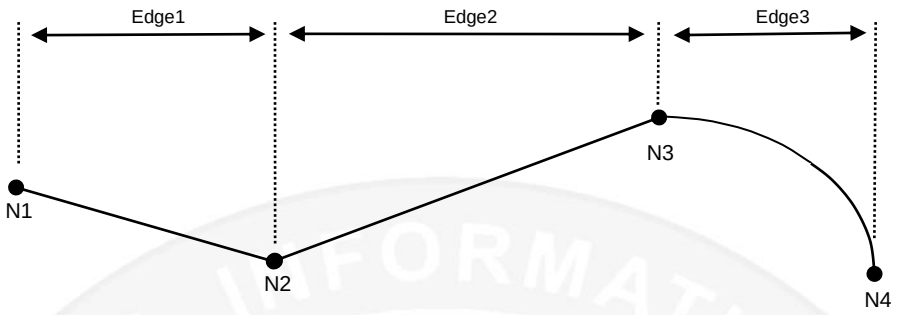
Curve C2a is a GM_OrietableCurve with a positive orientation forming the interior ring (R2) of Boundary A. An explanation of the coordinate geometry representing curves can be found in A.1.1.

Surface B is represented by the surface patch GM_Polygon (Patch B). Its GM_SurfaceBoundry (Boundary B) has one exterior ring (R3).

Curve C2b is a GM_OrientableCurve with a negative orientation forming the exterior ring R3.

A.2 Topology

A.2.1 Edge Example



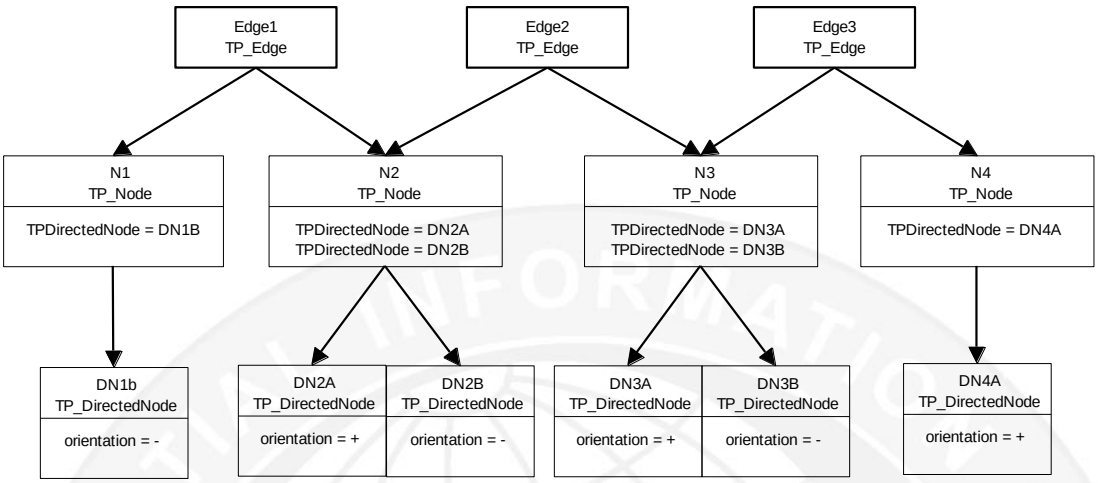


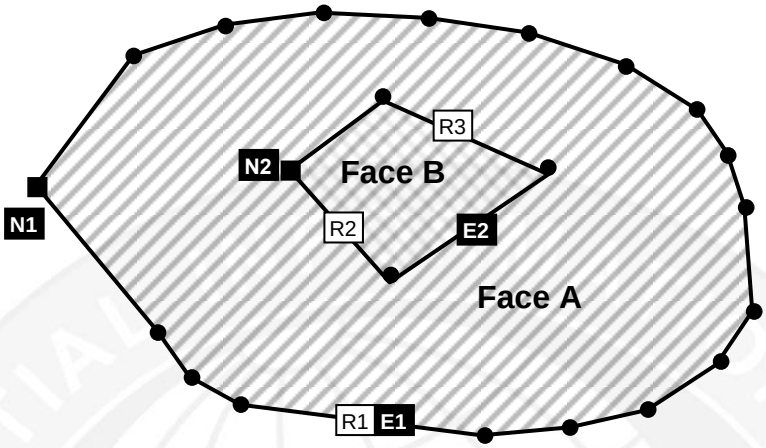
Figure A.3 – Edge Example

The following describes the basic principles of the topological elements of the edge example (Figure A.3).

TP_Edges E1,E2 and E3 each have TP_DirectedNodes which start and end the edge. TP_DirectedNodes can be described simplistically as the two halves of a TP_Node defining its positive and negative orientation. A negative TP_DirectedNode is at the start of the edge and a positive TP_DirectedNode is at the end of the edge.

In the next example the more complex relationship between TP_DirectedNodes and TP_DirectedEdges is demonstrated.

A.2.2 Face Example



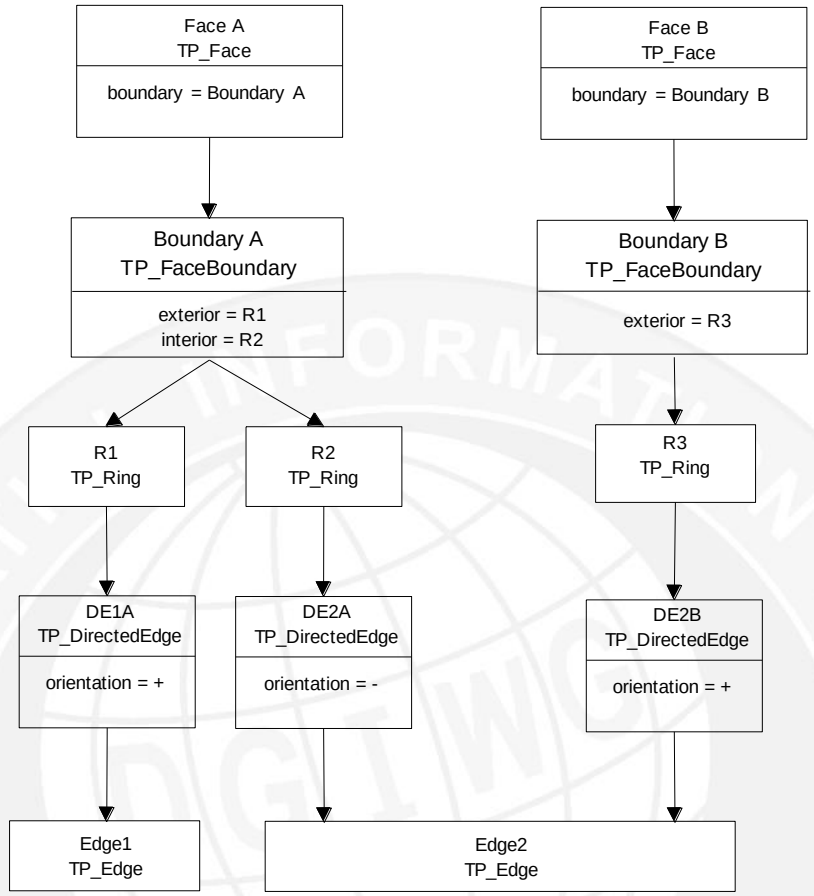


Figure A.4 – Face Example

The following describes the basic principles of the topological elements of the face example (Figure A.4).

Face A is represented by a TP_Face. Its boundary (Boundary A) is comprised of two TP_Rings, one exterior (R1) and one interior (R2). The rings in turn are comprised of TP_Edges which simplistically are comprised of two TP_DirectedEdges, one with a positive orientation and the other with a negative orientation.

Face B is represented by a TP_Face. Its boundary (Boundary B) is comprised of one exterior TP_Ring (R1).

Edge E1 forms the absolute external boundary of Face A and its directed edge E1a has a positive orientation forming the exterior ring R1.

Edge E2 forms both the interior ring R2 (directed node E2a with a negative orientation) of Face A and the exterior ring R3 (directed node E2b with a positive orientation) of Face B.

A.3 2.5 Dimensional Geometry

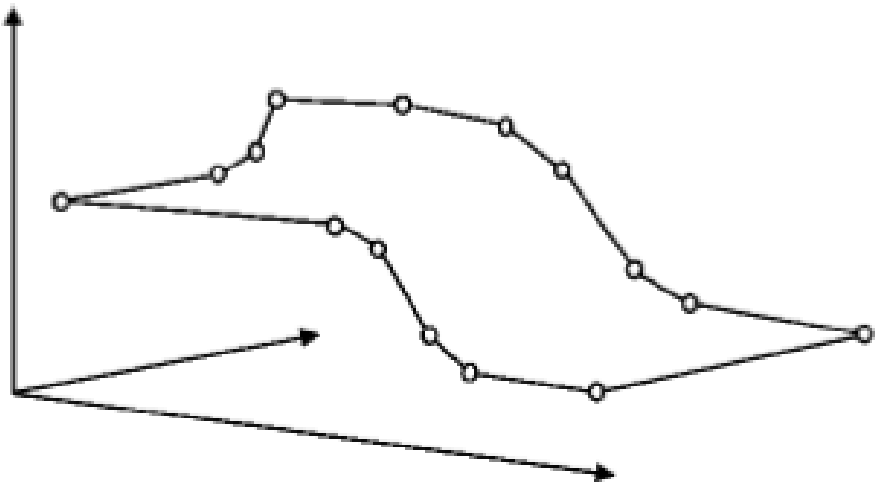


Figure A.5 – 2.5D Example

In the depicted example, the curve which constitutes the exterior boundary of the GM_Polygon consists of GM_LineStrings with 3D control points. Note that the surface interpolation must be “none”, which means that the position of interior points is not determined. The “planar” interpolation would be only acceptable if all points were lying on a plane.

A.4 Realization Example

The model defined in ISO 19107 is, in general, not an implementation model as it mostly consists of <<Type>> stereotyped classes. Types are interfaces with internal state. Just as interfaces, types are realized through implementation classes. The mapping need not be one to one. An implementation class can realize multiple types/interfaces and a type/interface can be realized by multiple implementation classes. The following example shows how the GM_Surface type and its boundary associations can be realized. Figure A.6 shows the types that define a surface and its boundary as specified in the Bounded 2D Surface Complex Profile.

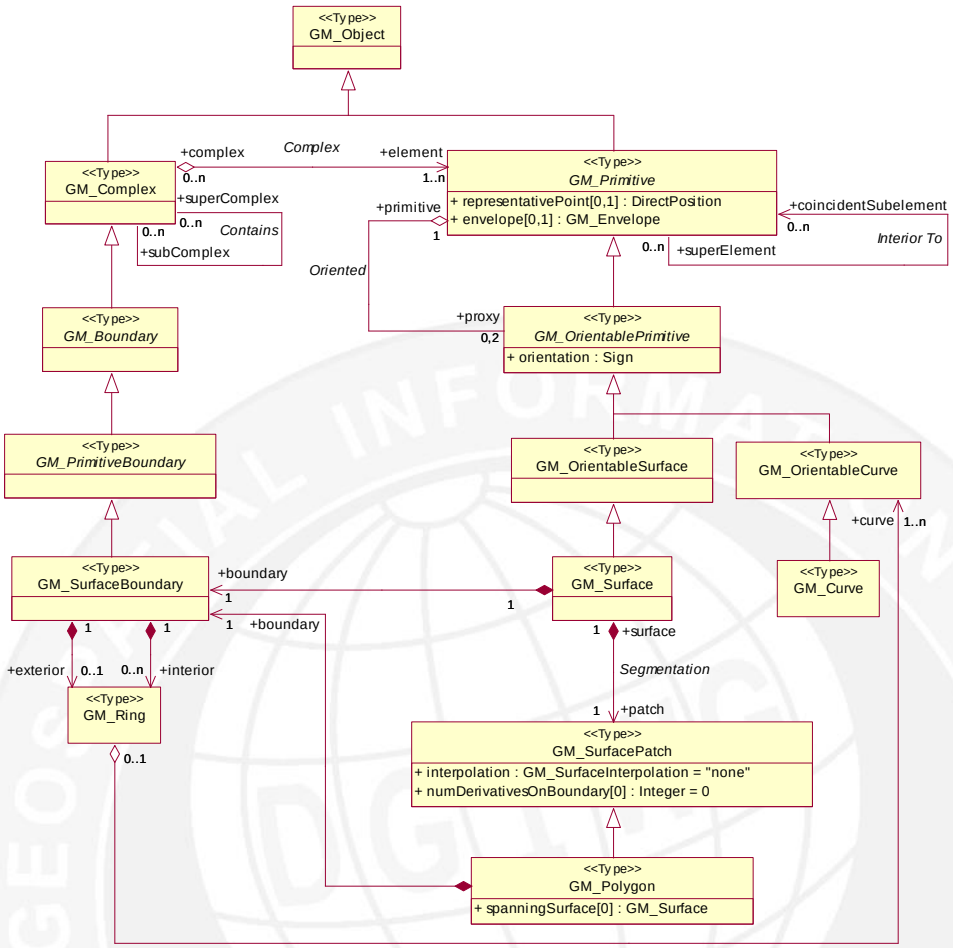


Figure A.6 – Surface and boundary defining types

The type hierarchy in Figure A.6 can be flattend as shown in Figure A.7. Certain associations have been omitted since they have no meaning in the flattened context. The Contains associtation that represents the containment of one complex within another is immaterial in this context as is the InteriorTo association. The Oriented association is not represented in this diagram but is represented in the realization (Figure A.8).

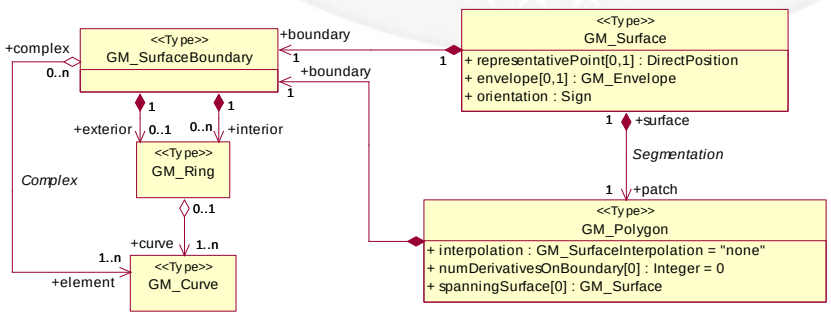


Figure A.7 – Flattened class hierarchy

Figure A.8 shows how these types can be realized. The focus is on the surface and its boundary associations. The implementation class Surface realizes GM_Surface, GM_Polygon and GM_SurfaceBoundary. The implementation class Ring realizes GM_Ring and Curve realizes GM_Curve.

The attributes of the implementation class Surface realize the internal state of the types which Surface realizes: GM_Surface, GM_Polygon and GM_SurfaceBoundary. The attribute GM_Surface::representativePoint is realized by the attribute Surface::focus. The attribute GM_Surface::envelope is realized by the attributes Surface::min and Surface::max. The attribute GM_Surface::orientation is implied "+". The attribute GM_Polygon::interpolation is implied "none". Neither the attribute GM_Polygon::numDerivativesOnBoundary nor the attribute GM_Polygon::spanningSurface is populated in the Bounded 2D Surface Complex Profile and does not appear in the realization.

The Segmentation association and the boundary roles are realized through the combined realization of the types GM_Surface, GM_Polygon and GM_SurfaceBoundary by the Surface implementation class. The ring associations that are specified by the roles GM_SurfaceBoundary::exterior and GM_SurfaceBoundary::interior are realized by the association role Surface::ring where the first ring is the exterior ring and any subsequent rings are interior. The association role GM_Ring::curve is realized by the Ring::curve role. The Complex association is realized by the combination of the Surface::ring and the Ring::curve roles. The Orientation association class realizes the orientation attributes in both GM_Surface and GM_Curve.

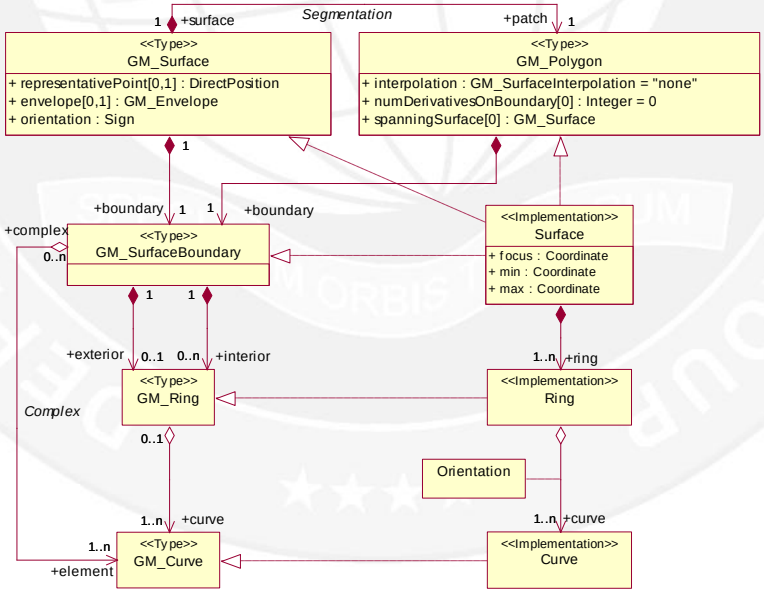


Figure A.8 – Realization of GM

By making the associations bidirectional the implementation classes also realize the TP types as in the Topological 2D Surface Complex Profile (Figure A.9). Most of the associations in TP mirror those in GM. By adding bidirecitionality the additional CoBoundary association is realized.

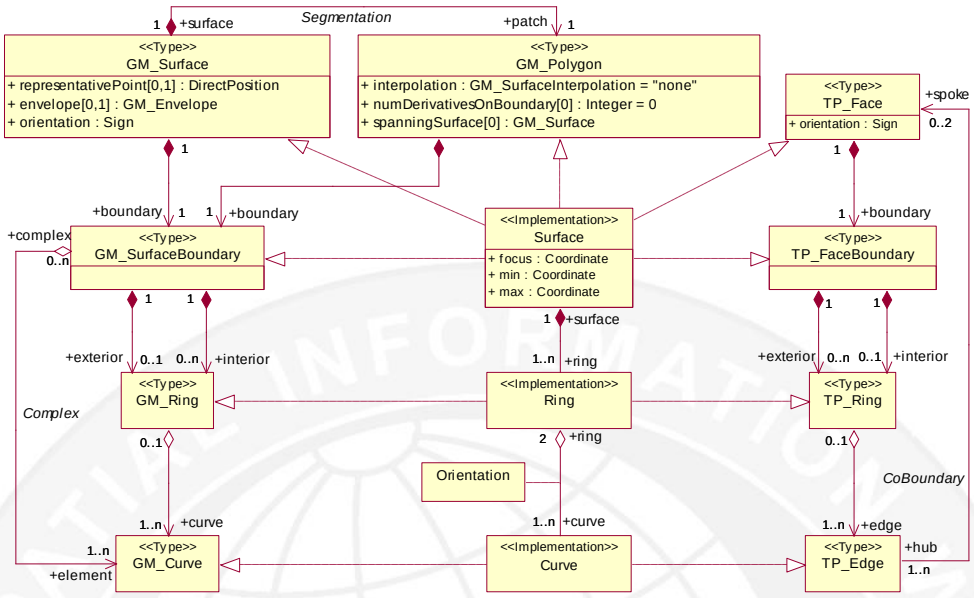


Figure A.9 – Realization of GM and TP

Annex B

(normative)

2 and 2.5D Sub-Profiles

B.1 Scope

The main body of this document specifies a general profile of 19107. This annex specifies six sub-profiles of the general profile. The general profile will be referred to as the parent profile in the remainder of this annex. These sub-profiles are characterized in the following table. For each profile, the minimum conformance class from Annex A of ISO 19107 is given, along with the VRF topology levels that correspond to each profile.

Topological Dimension (Minimum Coordinate Dimension)	2	
Maximum Geometric Primitive Dimension / Boundary & Co-Boundary Information	1 Curves	2 Surfaces
Collection of GM Primitives Primitive interiors not necessarily disjoint Not all boundary objects necessarily exist No GM Complex No TP objects	① 2D curve collection A.1.1.2 VRF Levels 0 & 1	② 2D surface collection A.1.1.3 19137
GM Primitives in a GM Complex Primitive interiors are disjoint All boundary objects exist All boundary associations exist No TP objects exist	③ Bounded 2D curve complex A.2.1.1	④ Bounded 2D surface complex A.2.1.2
TP Primitives in a TP Complex and GM Primitives in a GM Complex Primitive interiors are disjoint All boundary associations exist All co-boundary associations exist	⑤ Topological 2D curve complex A.4.1.1 VRF Level 2	⑥ Topological 2D surface complex A.4.1.2 VRF Level 3

Figure 1.1 – 2D and 2.5D Profile Characteristics

Each of these profiles describes a collection of geometric primitives (with, in some cases, corresponding topological primitives) that meet certain criteria. The six profiles are organized into three pairs. The first profile in each pair includes only 1D primitives (curves), while the second profile in each pair also includes 2D primitives (surfaces).

The first pair of profiles is distinguished from the others by the fact that the interiors of the geometric primitives in the collection are not guaranteed to be disjoint – i.e., primitives may

overlap or cross one another, or be duplicated. Nor are all of the primitives that bound other primitives necessarily included in the collection. Thus, the primitives do not form a geometric complex. The first of these two profiles includes only points and curves, corresponding to VRF Level 0. VRF Level 1 also corresponds to this profile, because although it requires each curve to have its bounding points, it allows curves to cross. The second profile in this pair includes surfaces, and is very similar to ISO 19137.

In the second pair of profiles, all primitive interiors are required to be disjoint, and all of the boundary primitives are included, so that all of the GM_Primitives form a geometric complex. Boundary associations exist between all geometric primitives and the primitives that make up their boundaries. The first of these profiles define a network of curves that form a complex, while the second defines a surface complex.

The third pair of profiles adds the corresponding TP_Primitives within a maximal TP_Complex, and includes all of the co-boundary associations. These two profiles correspond to VRF Levels 2 and 3, respectively.

B.2 Profiles

B.2.1 2D Curve Collection Profile

B.2.1.1 Introduction

This profile consists of an unconstrained collection of 0D and 1D geometric primitives.

B.2.1.2 Class Diagrams

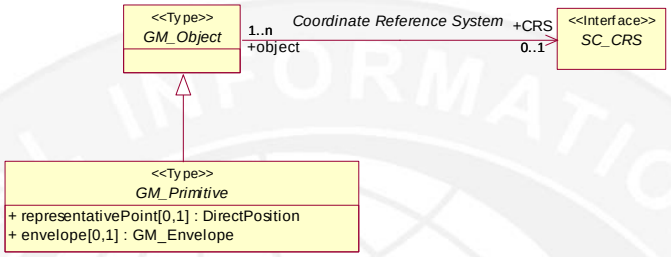


Figure B.2.1.1 – Geometry Object

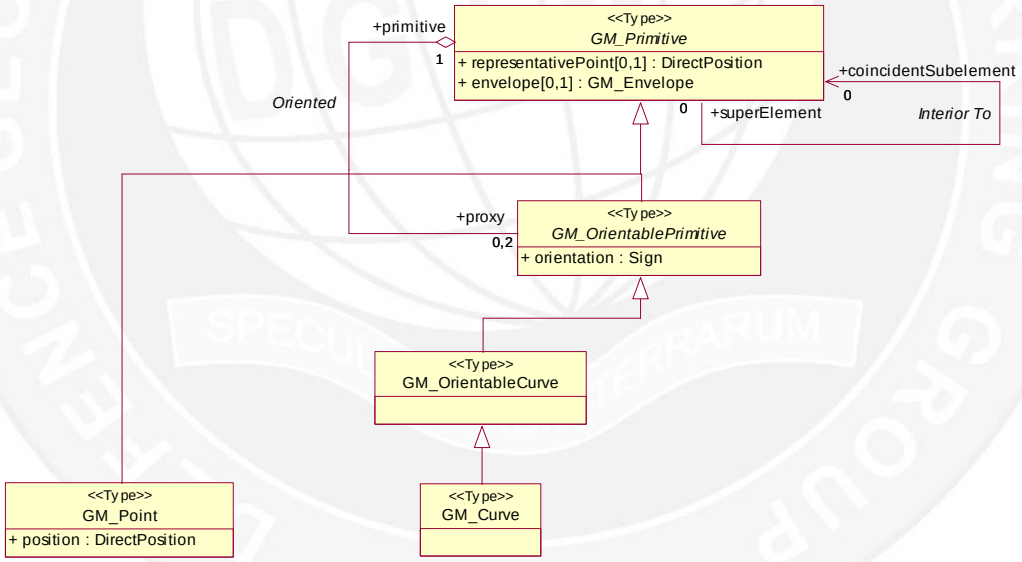


Figure B.2.1.2 – Geometry Primitive

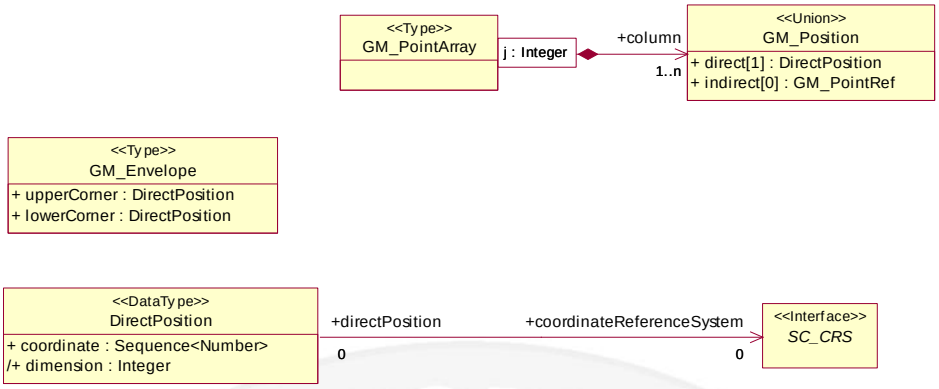


Figure B.2.1.3 – Coordinate Package

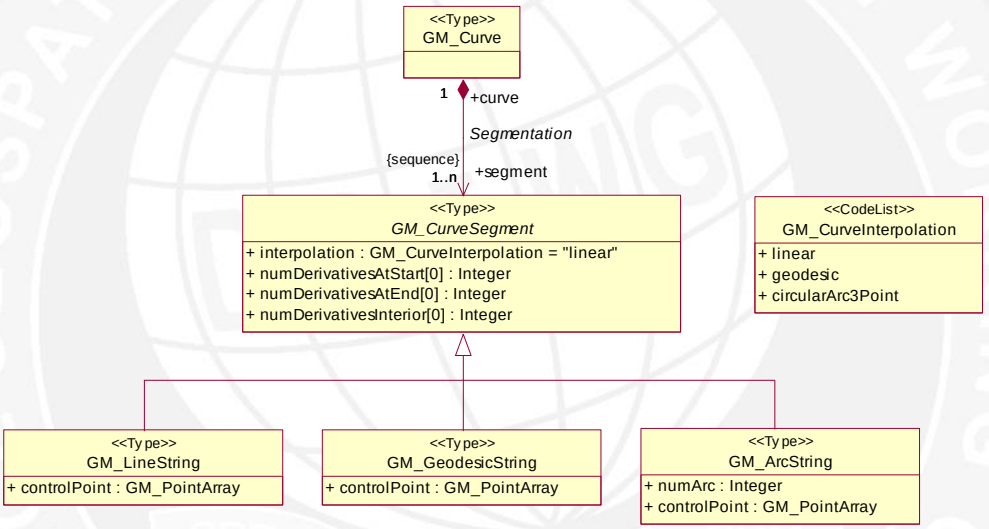


Figure B.2.1.4 – Curve Segment

B.2.1.3 Omitted Constructs

Classes from the parent profile are omitted unless explicitly included. Associations between a class and an omitted class are also omitted.

B.2.1.4 Included Constructs

The following is a list of all the included classes preceded by their ISO 19107 section number and their DGIWG Profile 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.

- 6.2.2/6.2.1 GM_Object – specializations B.2.1.5.1, B.2.1.5.2
- 6.3.10/6.3.1 GM_Primitive – specializations B.2.1.5.1, B.2.1.5.2, B.2.1.6.1

- 6.3.11/6.3.2 GM_Point
- 6.3.13/6.3.3 GM_OrientablePrimitive
- 6.3.14/6.3.4 GM_OrientableCurve
- 6.3.16/6.3.6 GM_Curve
- 6.4.1/6.4.2 DirectPosition – specialization B.2.1.6.3
- 6.4.3/6.4.5 GM_Envelope
- 6.4.5/6.4.3 GM_Position – specialization B.2.1.5.3
- 6.4.6/6.4.4 GM_PointArray
- 6.4.8/6.4.7 GM_CurveInterpolation – specialization B.2.1.5.4
- 6.4.9/6.4.6 GM_CurveSegment – specializations B.2.1.5.5, B.2.1.6.2
- 6.4.10/6.4.8 GM_LineString
- 6.4.12/6.4.9 GM_GeodesicString
- 6.4.14/6.4.10 GM_ArcString

B.2.1.5 Class Specialization

B.2.1.5.1 GM_Object, GM_Primitive

The type GM_Object operation representativePoint is changed to an optional derived type GM_Primitive attribute.

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

B.2.1.5.2 GM_Object, GM_Primitive

The type GM_Object operation envelope is changed to an optional derived type GM_Primitive attribute.

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

B.2.1.5.3 GM_Position

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])
```

B.2.1.5.4 GM_CurveInterpolation

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

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

B.2.1.5.5 GM_CurveSegment

The multiplicity of all three GM_CurveSegment::numDerivatives attributes are restricted 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])
```

B.2.1.6 Association Specialization

B.2.1.6.1 InteriorTo

The multiplicity of the association roles “coincidentSubelement” and “superElement” of GM_Primitive is restricted from [0..n] to [0].

```
GM_Primitive::coincidentSubelement[0]                    :
Reference<GM_Primitive>                                  ([0..n])
GM_Primitive::superElement[0]     :     Reference<GM_Primitive>
                                                         ([0..n])
```

B.2.1.6.2 Segmentation

The multiplicity of the association role “curve” of GM_CurveSegment is restricted from [0,1] to [1].

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

B.2.1.6.3 Coordinate Reference System

The multiplicity of the association Coordinate Reference System of DirectPosition is restricted from [0,1] to [0].

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

B.2.2 2D Surface Collection Profile

B.2.2.1 Introduction

This profile consists of an unconstrained collection of 0D, 1D, and 2D geometric primitives.

B.2.2.2 Class Diagrams

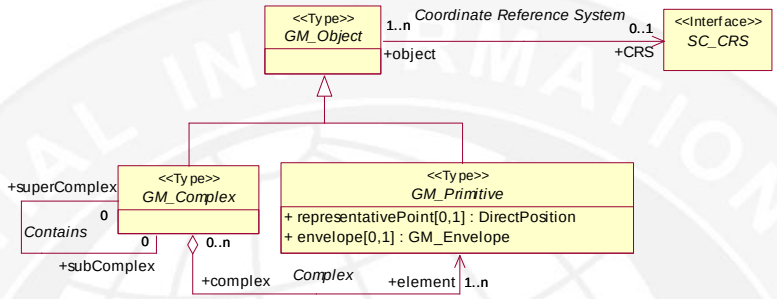


Figure B.2.2.1 – Geometry Object

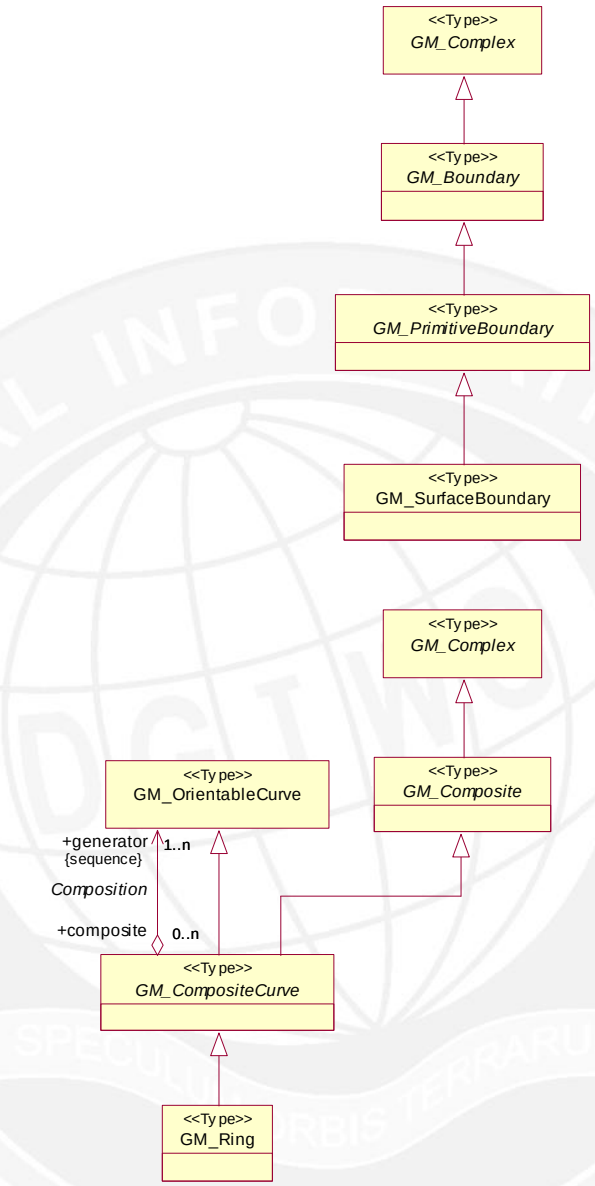


Figure B.2.2.2 – Geometry Boundary

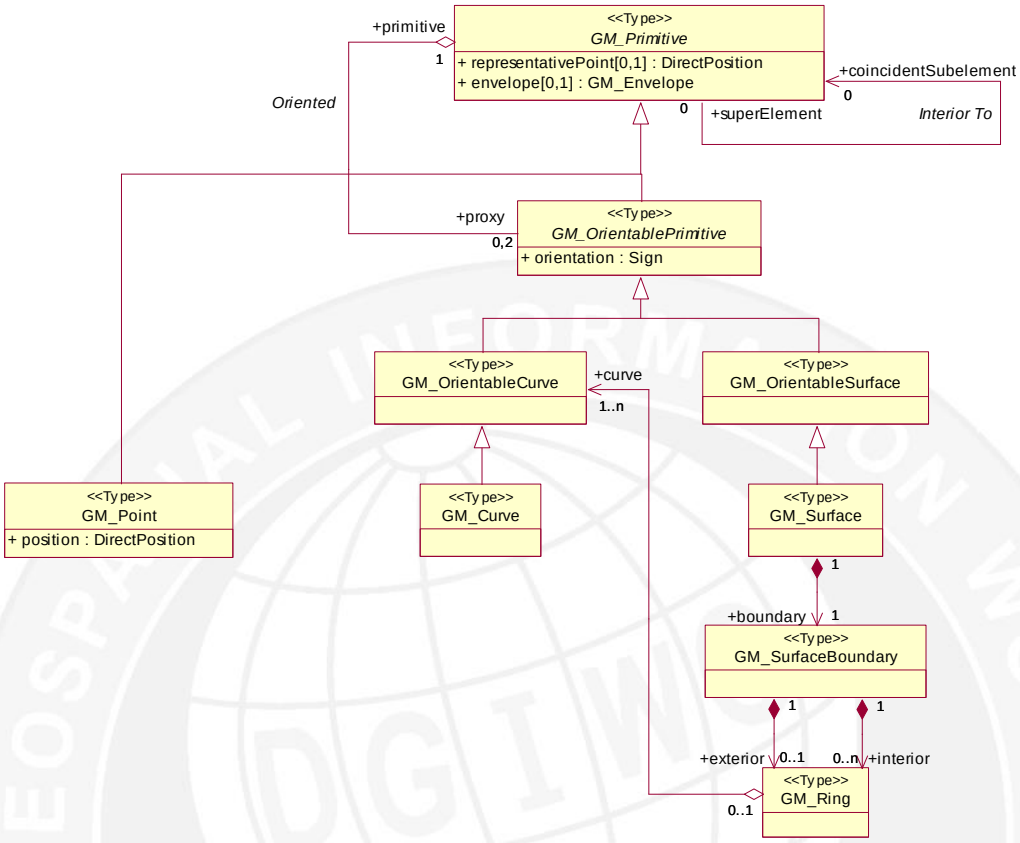


Figure B.2.2.3 – Geometry Primitive

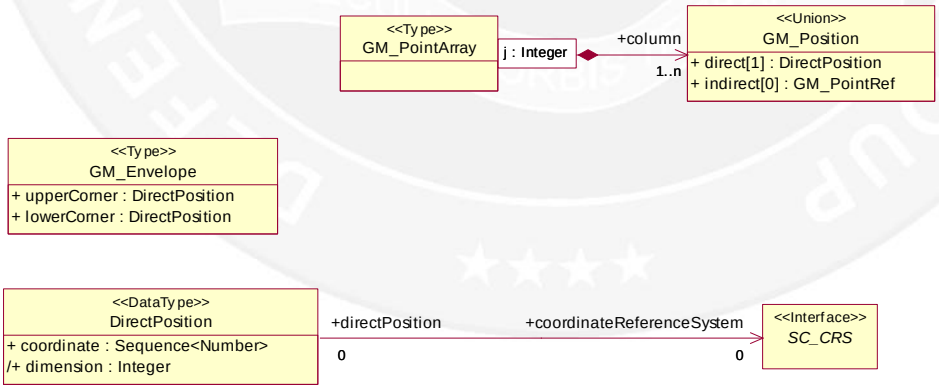


Figure B.2.2.4 – Coordinate Package

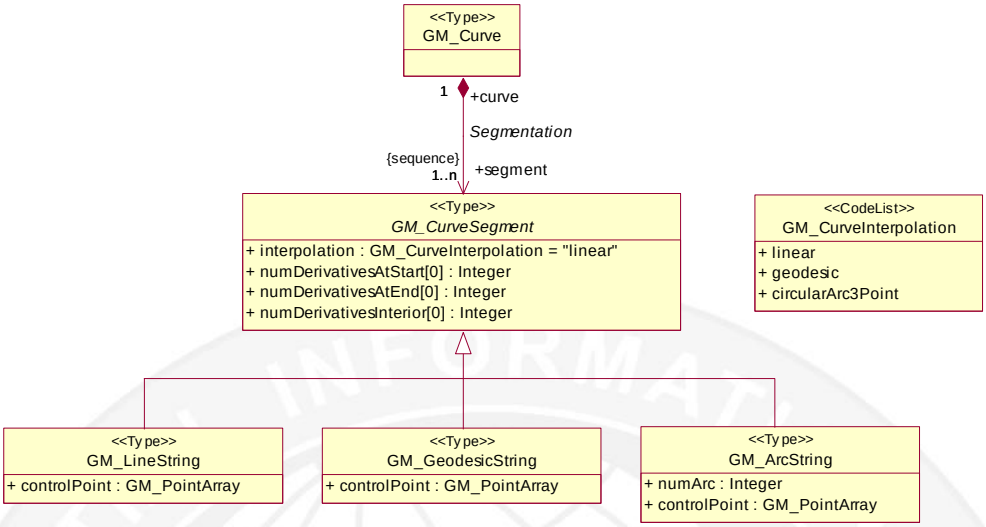


Figure B.2.2.5 – Curve Segment

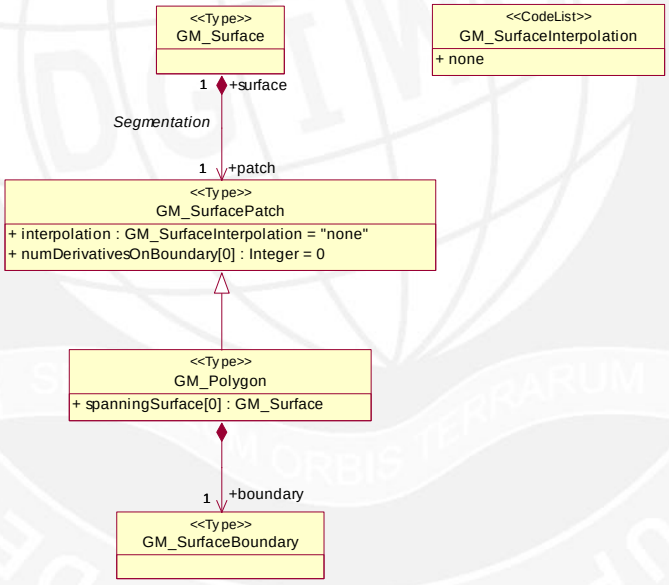


Figure B.2.2.6 – Surface Patch

B.2.2.3 Omitted Constructs

Classes from the parent profile are omitted unless explicitly included. Associations between a class and an omitted class are also omitted.

B.2.2.4 Included Constructs

The following is a list of all the included classes preceded by their ISO 19107 section number and their DGIWG Profile 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.

- 6.2.2/6.2.1 GM_Object – specializations B.2.2.5.1, B.2.2.5.2
- 6.3.2/6.3.8 GM_Boundary
- 6.3.4/6.3.9 GM_PrimitiveBoundary
- 6.3.6/6.3.12 GM_Ring
- 6.3.7/6.3.11 GM_SurfaceBoundary
- 6.3.10/6.3.1 GM_Primitive – specializations B.2.2.5.1, B.2.2.5.2, B.2.2.6.1
- 6.3.11/6.3.2 GM_Point
- 6.3.13/6.3.3 GM_OrientablePrimitive
- 6.3.14/6.3.4 GM_OrientableCurve
- 6.3.15/6.3.5 GM_OrientableSurface – specialization B.2.2.5.3
- 6.3.16/6.3.6 GM_Curve
- 6.3.17/6.3.7 GM_Surface – specializations B.2.2.5.3, B.2.2.6.3
- 6.4.1/6.4.2 DirectPosition – specialization B.2.2.6.4
- 6.4.3/6.4.5 GM_Envelope
- 6.4.5/6.4.3 GM_Position – specialization B.2.2.5.4
- 6.4.6/6.4.4 GM_PointArray
- 6.4.8/6.4.7 GM_CurveInterpolation – specialization B.2.2.5.5
- 6.4.9/6.4.6 GM_CurveSegment – specializations B.2.2.5.6, B.2.2.6.2
- 6.4.10/6.4.8 GM_LineString
- 6.4.12/6.4.9 GM_GeodesicString
- 6.4.14/6.4.10 GM_ArcString
- 6.4.32/6.4.13 GM_SurfaceInterpolation – specialization B.2.2.5.7
- 6.4.34/6.4.12 GM_SurfacePatch – specializations B.2.2.5.8, B.2.2.6.3
- 6.4.36/6.4.14 GM_Polygon – specialization B.2.2.5.9
- 6.6.2/6.4.15 GM_Complex – specializations B.2.2.5.10, B.2.2.6.5
- 6.6.3/6.4.16 GM_Composite
- 6.6.5/6.4.17 GM_CompositeCurve – specialization B.2.2.5.11
-

B.2.2.5 Class Specialization

B.2.2.5.1 GM_Object, GM_Primitive

The type GM_Object operation representativePoint is changed to an optional derived type GM_Primitive attribute.

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

B.2.2.5.2 GM_Object, GM_Primitive

The type GM_Object operation envelope is changed to an optional derived type GM_Primitive attribute.

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

B.2.2.5.3 GM_OrientableSurface, GM_Surface

The type GM_OrientableSurface operation boundary is changed to a derived type GM_Surface composition association to a GM_SurfaceBoundary object.

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

B.2.2.5.4 GM_Position

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])
```

B.2.2.5.5 GM_CurveInterpolation

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

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

B.2.2.5.6 GM_CurveSegment

The multiplicity of all three GM_CurveSegment::numDerivatives attributes are restricted 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])
```

B.2.2.5.7 GM_SurfaceInterpolation

The code list GM_SurfaceInterpolation is constrained to the value “none”.

```
GM_SurfaceInterpolation::

      none
```

B.2.2.5.8 GM_SurfacePatch

The multiplicity of the GM_SurfacePatch::numDerivativesOnBoundary attributes is restricted from [0,1] to [0].

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

B.2.2.5.9 GM_Polygon

The multiplicity of the GM_Polygon::spanningSurface attributes is restricted from [0,1] to [0].

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

B.2.2.5.10 GM_Complex

The class GM_Complex is changed from concrete to abstract thus limiting GM_Complex to the sub-types GM_Ring and GM_SurfaceBoundary.

B.2.2.5.11 GM_CompositeCurve

The class GM_CompositeCurve is changed from concrete to abstract thus limiting GM_CompositeCurve to the sub-type GM_Ring.

B.2.2.6 Association Specialization

B.2.2.6.1 InteriorTo

The multiplicity of the association roles “coincidentSubelement” and “superElement” of GM_Primitive is restricted from [0..n] to [0].

```
GM_Primitive::coincidentSubelement[0]                    :
Reference<GM_Primitive>                                  ([0..n])

GM_Primitive::superElement[0]     :     Reference<GM_Primitive>
                                                         ([0..n])
```

B.2.2.6.2 Segmentation

The multiplicity of the association role “curve” of GM_CurveSegment is restricted from [0,1] to [1].

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

B.2.2.6.3 Segmentation

The multiplicity of the association role “patch” of GM_Surface is restricted from [1..n] to [1]. The multiplicity of the association role “surface” of GM_SurfacePatch is restricted from [0,1] to [1].

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

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

B.2.2.6.4 Coordinate Reference System

The multiplicity of the association Coordinate Reference System of DirectPosition is restricted from [0,1] to [0].

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

B.2.2.6.5 Contains

The multiplicity of the association roles “subComplex” and “superComplex” of GM_Complex is restricted from [0..n] to [0].

```
GM_Complex::subComplex[0] : GM_Complex                ([0..n])

GM_Complex::superComplex[0] : GM_Complex              ([0..n])
```

B.2.3 Bounded 2D Curve Complex Profile

B.2.3.1 Introduction

This profile consists of a collection of 0D and 1D geometric primitives that are capable of being contained within a single maximal geometric complex. In accordance with the definition of a geometric complex, the interiors of all of the primitives are required to be disjoint, and the collection of primitives is complete, in that for any given primitive in the collection, the primitives that make up its boundary are also included. Thus, this collection forms a network of curves that meet at common points.

B.2.3.2 Class Diagrams

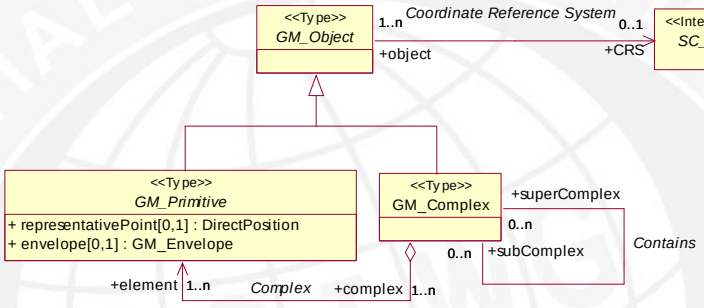


Figure B.2.3.1 – Geometry Object

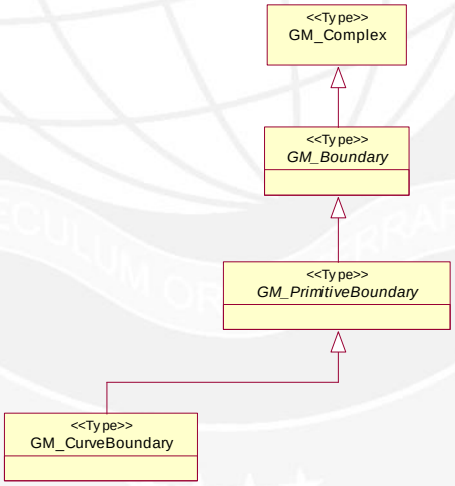


Figure B.2.3.2 – Geometry Boundary

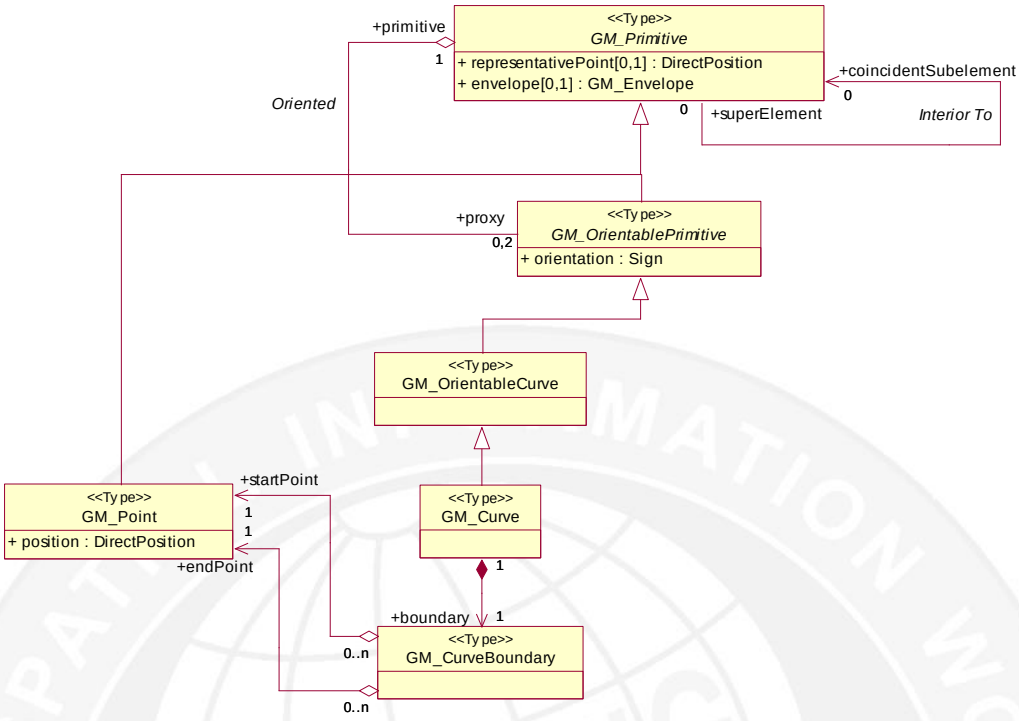


Figure B.2.3.3 – Geometry Primitive

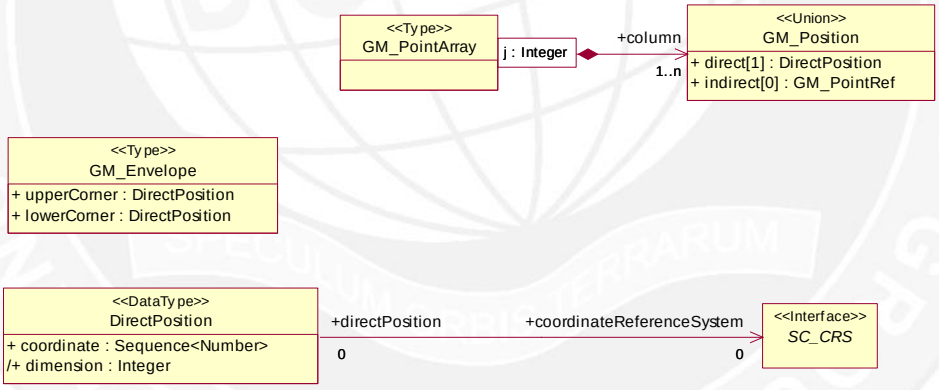


Figure B.2.3.4 – Coordinate Package

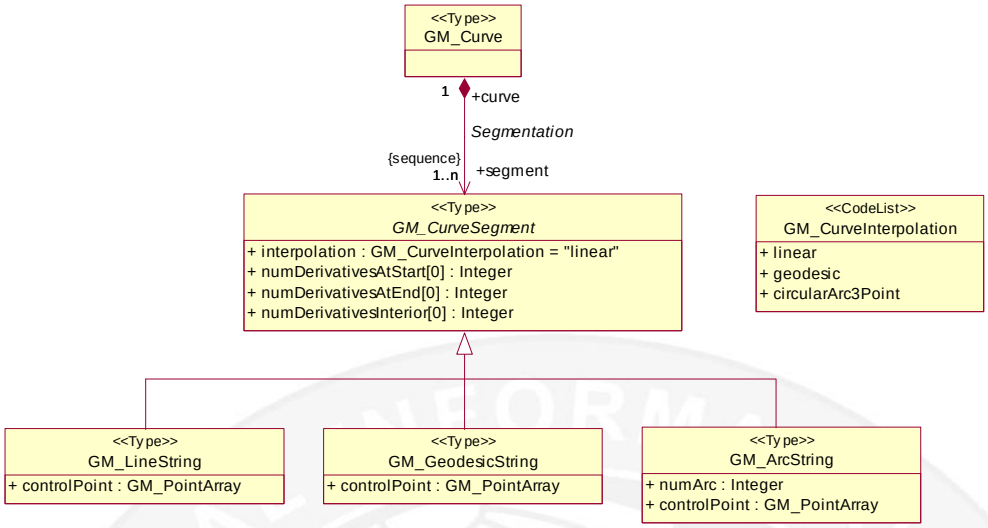


Figure B.2.3.5 – Curve Segment

B.2.3.3 Omitted Constructs

Classes from the parent profile are omitted unless explicitly included. Associations between a class and an omitted class are also omitted.

B.2.3.4 Included Constructs

The following is a list of all the included classes preceded by their ISO 19107 section number and their DGIWG Profile 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.

- 6.2.2/6.2.1 GM_Object – specializations B.2.3.5.1, B.2.3.5.2
- 6.3.2/6.3.8 GM_Boundary
- 6.3.4/6.3.9 GM_PrimitiveBoundary
- 6.3.5/6.3.10 GM_CurveBoundary
- 6.3.10/6.3.1 GM_Primitive – specializations B.2.3.5.1, B.2.3.5.2, B.2.3.6.1, B.2.3.6.2
- 6.3.11/6.3.2 GM_Point
- 6.3.13/6.3.3 GM_OrientablePrimitive
- 6.3.14/6.3.4 GM_OrientableCurve – specialization B.2.3.5.3
- 6.3.16/6.3.6 GM_Curve – specialization B.2.3.5.3
- 6.4.1/6.4.2 DirectPosition – specialization B.2.3.6.4

- 6.4.3/6.4.5 GM_Envelope
- 6.4.5/6.4.3 GM_Position – specialization B.2.3.5.4
- 6.4.6/6.4.4 GM_PointArray
- 6.4.8/6.4.7 GM_CurveInterpolation – specialization B.2.3.5.5
- 6.4.9/6.4.6 GM_CurveSegment – specializations B.2.3.5.6, B.2.3.6.3
- 6.4.10/6.4.8 GM_LineString
- 6.4.12/6.4.9 GM_GeodesicString
- 6.4.14/6.4.10 GM_ArcString
- 6.6.2/6.4.15 GM_Complex

B.2.3.5 Class Specialization

B.2.3.5.1 GM_Object, GM_Primitive

The type GM_Object operation representativePoint is changed to an optional derived type GM_Primitive attribute.

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

B.2.3.5.2 GM_Object, GM_Primitive

The type GM_Object operation envelope is changed to an optional derived type GM_Primitive attribute.

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

B.2.3.5.3 GM_OrientableCurve, GM_Curve

The type GM_OrientableCurve operation boundary is changed to a derived type GM_Curve composition association to a GM_CurveBoundary object.

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

B.2.3.5.4 GM_Position

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])
```

B.2.3.5.5 GM_CurveInterpolation

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

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

B.2.3.5.6 GM_CurveSegment

The multiplicity of all three GM_CurveSegment::numDerivatives attributes are restricted 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])
```

B.2.3.6 Association Specialization

B.2.3.6.1 InteriorTo

The multiplicity of the association roles “coincidentSubelement” and “superElement” of GM_Primitive is restricted from [0..n] to [0].

```
GM_Primitive::coincidentSubelement[0]                          :
Reference<GM_Primitive>                                    ([0..n])
GM_Primitive::superElement[0]     :     Reference<GM_Primitive>
                                                           ([0..n])
```

B.2.3.6.2 Complex

The multiplicity of the association role “complex” of GM_Primitive is restricted from [0..n] to [1..n] and a GM_Primitive has exactly one maximal complex.

```
GM_Primitive::complex[1..n] : Reference<GM_Complex> ([0..n])
context GM_Primitive inv:
```

self.maximalComplex()->size = 1

B.2.3.6.3 Segmentation

The multiplicity of the association role “curve” of GM_CurveSegment is restricted from [0,1] to [1].

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

B.2.3.6.4 Coordinate Reference System

The multiplicity of the association Coordinate Reference System of DirectPosition is restricted from [0,1] to [0].

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

B.2.4 Bounded 2D Surface Complex Profile

B.2.4.1 Introduction

This profile consists of a collection of 0D, 1D, and 2D geometric primitives that are capable of being contained within a single maximal geometric complex. In accordance with the definition of a geometric complex, the interiors of all of the primitives are required to be disjoint, and the collection of primitives is complete, in that for any given primitive in the collection, the primitives that make up its boundary are also included. Thus, this collection forms a single composite surface.

B.2.4.2 Class Diagrams

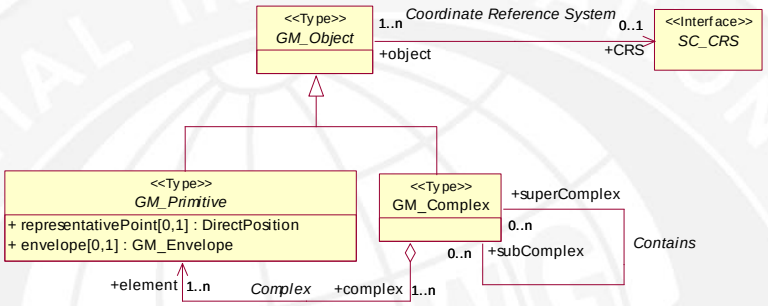


Figure B.2.4.1 – Geometry Object

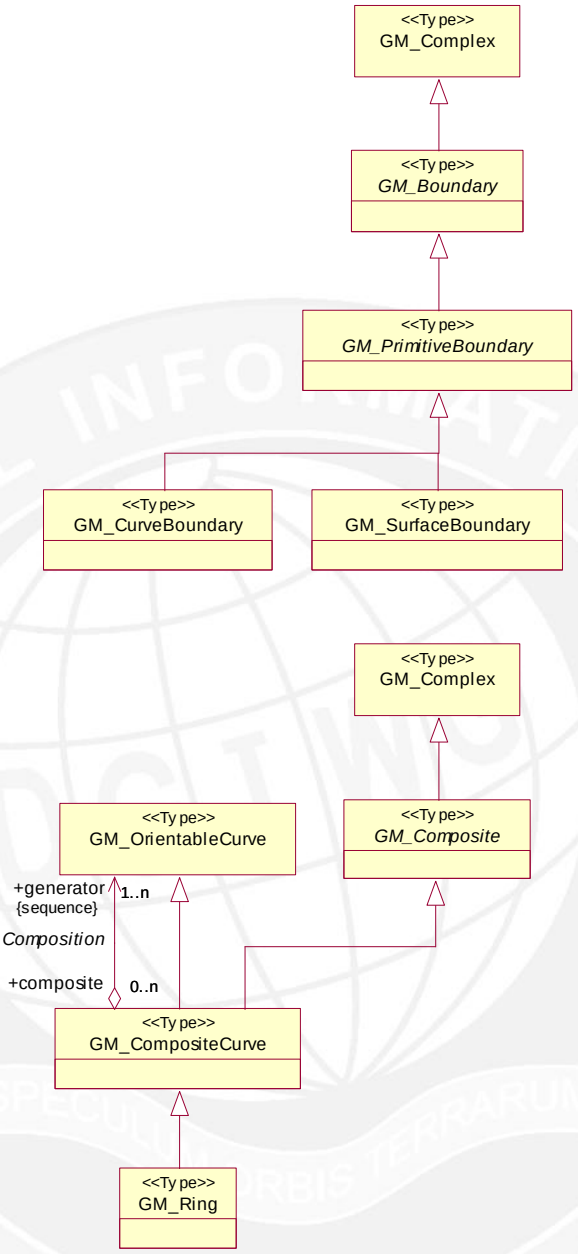


Figure B.2.4.2 – Geometry Boundary

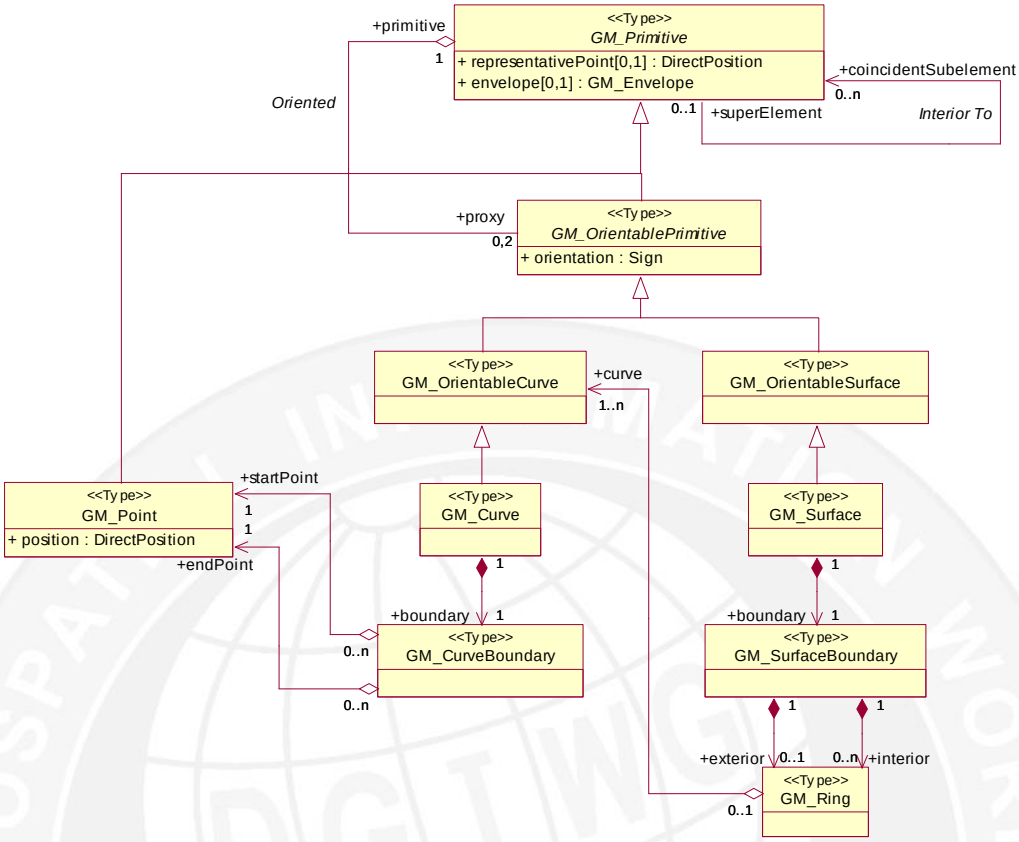


Figure B.2.4.3 – Geometry Primitive

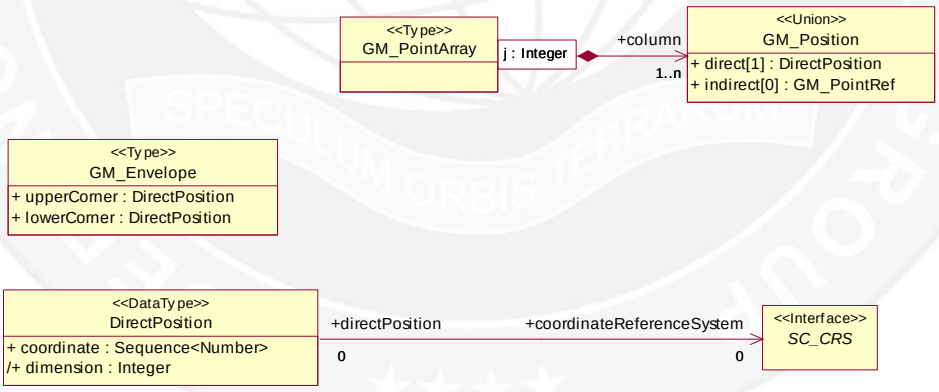


Figure B.2.4.4 – Coordinate Package

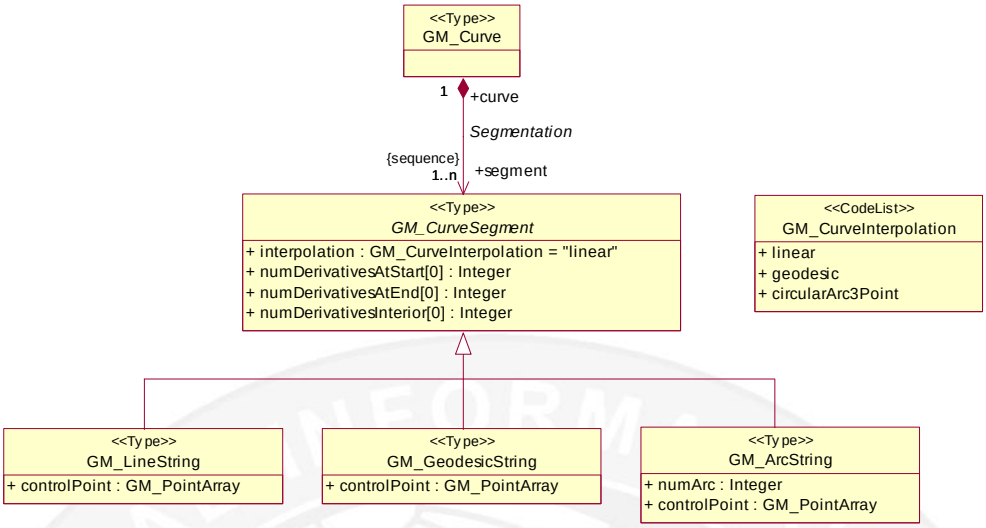


Figure B.2.4.5 – Curve Segment

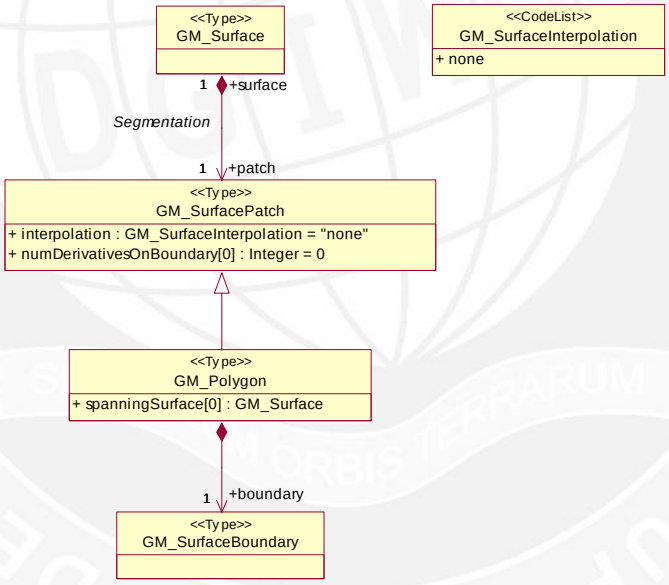


Figure B.2.4.6 – Surface Patch

B.2.4.3 Omitted Constructs

Classes from the parent profile are omitted unless explicitly included. Associations between a class and an omitted class are also omitted.

B.2.4.4 Included Constructs

The following is a list of all the included classes preceded by their ISO 19107 section number and their DGIWG Profile 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.

- 6.2.2/6.2.1 GM_Object – specializations B.2.4.5.1, B.2.4.5.2
- 6.3.2/6.3.8 GM_Boundary
- 6.3.4/6.3.9 GM_PrimitiveBoundary
- 6.3.5/6.3.10 GM_CurveBoundary
- 6.3.6/6.3.12 GM_Ring
- 6.3.7/6.3.11 GM_SurfaceBoundary
- 6.3.10/6.3.1 GM_Primitive – specializations B.2.4.5.1, B.2.4.5.2, B.2.4.6.1, B.2.4.6.2
- 6.3.11/6.3.2 GM_Point
- 6.3.13/6.3.3 GM_OrientablePrimitive
- 6.3.14/6.3.4 GM_OrientableCurve – specialization B.2.4.5.3
- 6.3.15/6.3.5 GM_OrientableSurface – specialization B.2.4.5.4
- 6.3.16/6.3.6 GM_Curve – specialization B.2.4.5.3
- 6.3.17/6.3.7 GM_Surface – specializations B.2.4.5.4, B.2.4.6.4
- 6.4.1/6.4.2 DirectPosition – specialization B.2.4.6.5
- 6.4.3/6.4.5 GM_Envelope
- 6.4.5/6.4.3 GM_Position – specialization B.2.4.5.5
- 6.4.6/6.4.4 GM_PointArray
- 6.4.8/6.4.7 GM_CurveInterpolation – specialization B.2.4.5.6
- 6.4.9/6.4.6 GM_CurveSegment – specializations B.2.4.5.7, B.2.4.6.3
- 6.4.10/6.4.8 GM_LineString
- 6.4.12/6.4.9 GM_GeodesicString
- 6.4.14/6.4.10 GM_ArcString
- 6.4.32/6.4.13 GM_SurfaceInterpolation – specialization B.2.4.5.8
- 6.4.34/6.4.12 GM_SurfacePatch – specializations B.2.4.5.9, B.2.4.6.4
- 6.4.36/6.4.14 GM_Polygon – specialization B.2.4.5.10
- 6.6.2/6.4.15 GM_Complex

- 6.6.3/6.4.16 GM_Composite
- 6.6.5/6.4.17 GM_CompositeCurve

B.2.4.5 Class Specialization

B.2.4.5.1 GM_Object, GM_Primitive

The type GM_Object operation representativePoint is changed to an optional derived type GM_Primitive attribute.

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

B.2.4.5.2 GM_Object, GM_Primitive

The type GM_Object operation envelope is changed to an optional derived type GM_Primitive attribute.

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

B.2.4.5.3 GM_OrientableCurve, GM_Curve

The type GM_OrientableCurve operation boundary is changed to a derived type GM_Curve composition association to a GM_CurveBoundary object.

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

B.2.4.5.4 GM_OrientableSurface, GM_Surface

The type GM_OrientableSurface operation boundary is changed to a derived type GM_Surface composition association to a GM_SurfaceBoundary object.

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

B.2.4.5.5 GM_Position

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])
```

B.2.4.5.6 GM_CurveInterpolation

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

```
GM_CurveInterpolation::

     linear

     geodesic

     circularArc3Points
```

B.2.4.5.7 GM_CurveSegment

The multiplicity of all three GM_CurveSegment::numDerivatives attributes are restricted 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])
```

B.2.4.5.8 GM_SurfaceInterpolation

The code list GM_SurfaceInterpolation is constrained to the value “none”.

```
GM_SurfaceInterpolation::

     none
```

B.2.4.5.9 GM_SurfacePatch

The multiplicity of the GM_SurfacePatch::numDerivativesOnBoundary attributes is restricted from [0,1] to [0].

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

B.2.4.5.10 GM_Polygon

The multiplicity of the GM_Polygon::spanningSurface attributes is restricted from [0,1] to [0].

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

B.2.4.6 Association Specialization

B.2.4.6.1 InteriorTo

The InteriorTo association only associates GM_Primitive with a dimensional difference greater than one. The multiplicity of the association role “superElement” of GM_Primitive is restricted from [0..n] to [0..1]. The association is only navigable from super-element to coincident sub-element.

```
GM_Primitive::superElement[0..1]   :   Reference<GM_Primitive>
                                                    ([0..n])

context GM_Primitive inv:

     coincidentSubelement->forAll( sub |

          sub.dimension() < self.dimension() – 1 )
```

B.2.4.6.2 Complex

The multiplicity of the association role “complex” of GM_Primitive is restricted from [0..n] to [1..n] and a GM_Primitive has exactly one maximal complex.

```
GM_Primitive::complex[1..n] : Reference<GM_Complex> ([0..n])

context GM_Primitive inv:

     self.maximalComplex()->size = 1
```

B.2.4.6.3 Segmentation

The multiplicity of the association role “curve” of GM_CurveSegment is restricted from [0,1] to [1].

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

B.2.4.6.4 Segmentation

The multiplicity of the association role “patch” of GM_Surface is restricted from [1..n] to [1]. The multiplicity of the association role “surface” of GM_SurfacePatch is restricted from [0,1] to [1].

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

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

B.2.4.6.5 Coordinate Reference System

The multiplicity of the association Coordinate Reference System of DirectPosition is restricted from [0,1] to [0].

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

B.2.5 Topological 2D Curve Complex Profile

B.2.5.1 Introduction

This profile consists of a collection of 0D and 1D topological primitives that are capable of being contained within a single maximal topological complex. A corresponding geometric primitive realizes each of the individual topological primitives, and a corresponding geometric complex realizes each topological complex. All boundary and co-boundary relationships are explicitly represented. Thus, the collection of primitives makes up a fully connected network of curves.

B.2.5.2 Class Diagrams

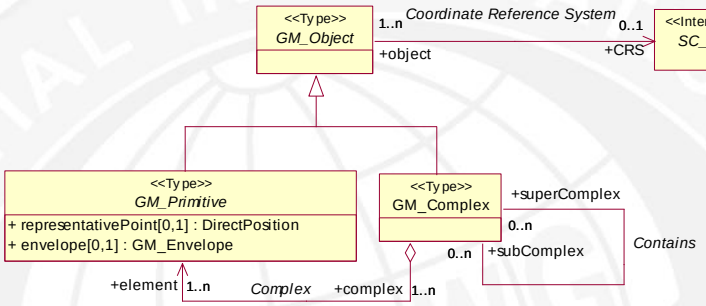


Figure B.2.5.1 – Geometry Object

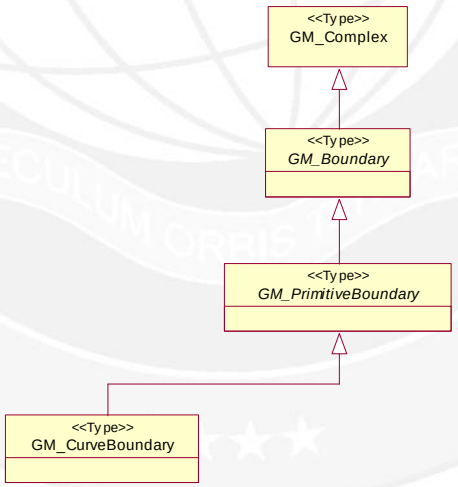


Figure B.2.5.2 – Geometry Boundary

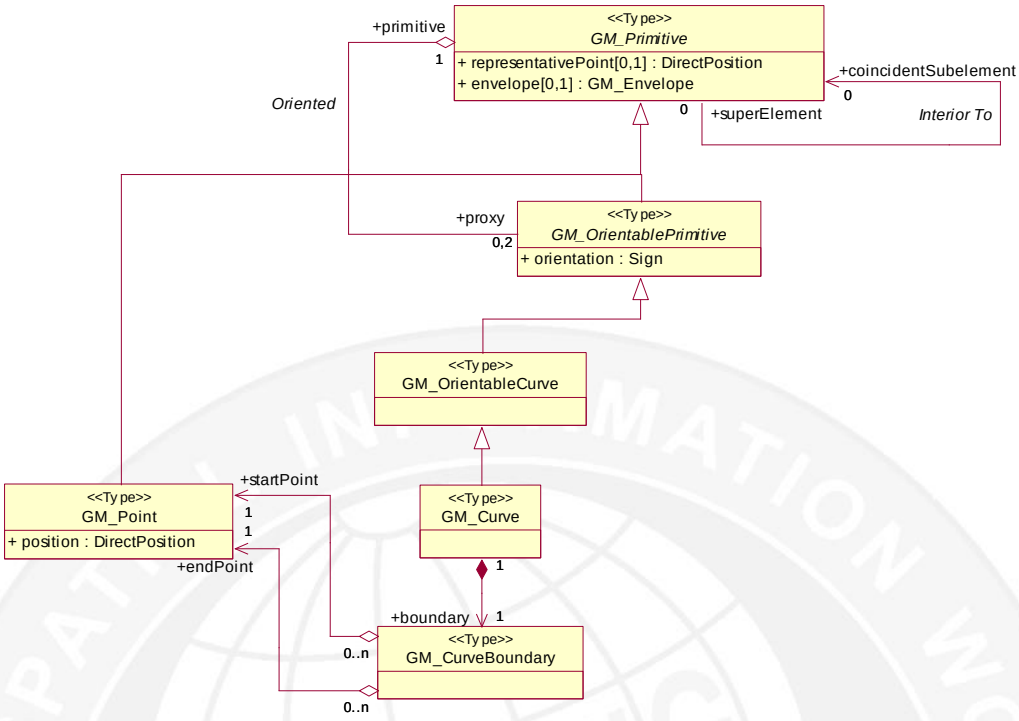


Figure B.2.5.3 – Geometry Primitive

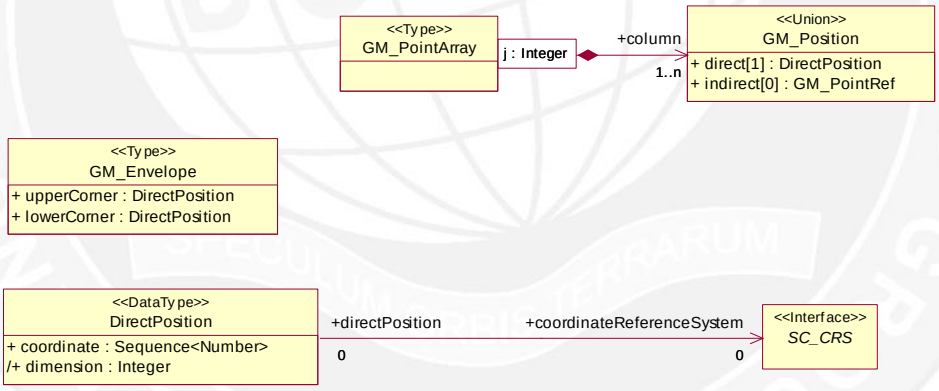


Figure B.2.5.4 – Coordinate Package

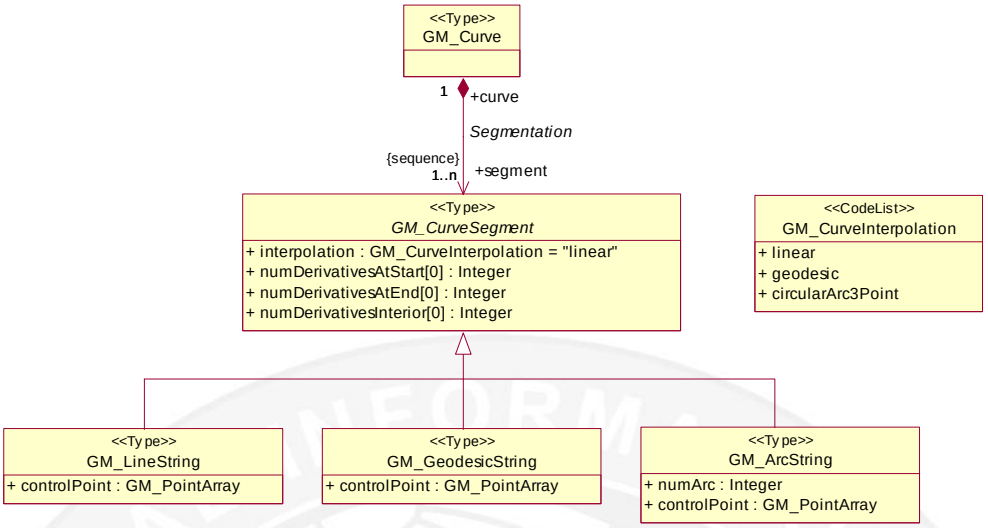


Figure B.2.5.5 – Curve Segment

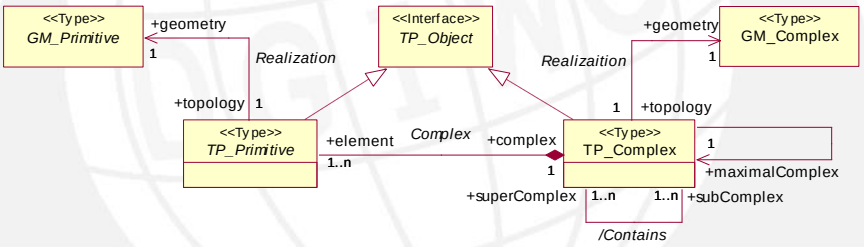


Figure B.2.5.6 – Topology Object

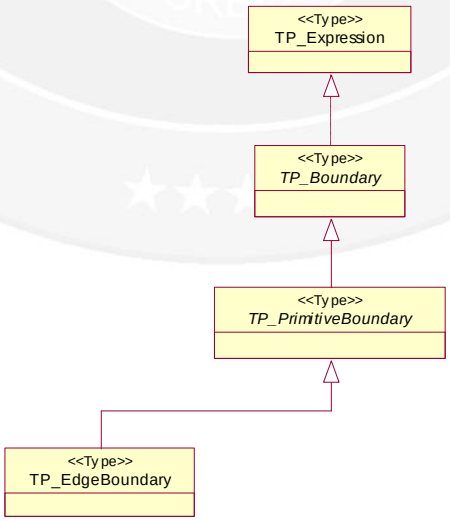


Figure B.2.5.7 – Topology Boundary

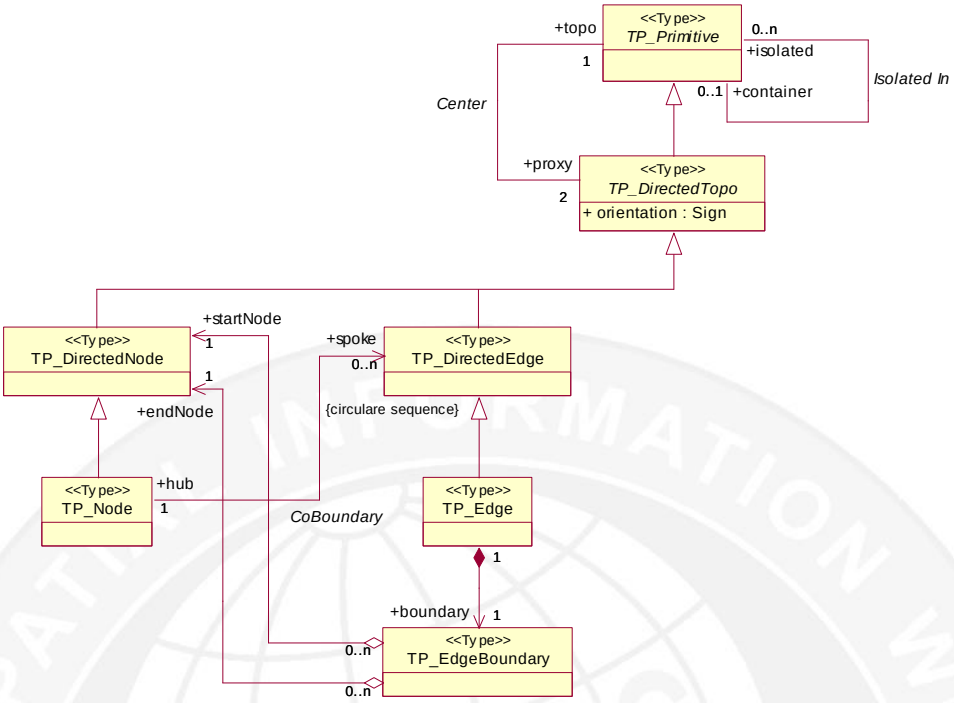


Figure B.2.5.8 – Topology Primitive

B.2.5.3 Omitted Constructs

Classes from the parent profile are omitted unless explicitly included. Associations between a class and an omitted class are also omitted.

B.2.5.4 Included Constructs

The following is a list of all the included classes preceded by their ISO 19107 section number and their DGIWG Profile 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.

- 6.2.2/6.2.1 GM_Object – specializations B.2.5.5.1, B.2.5.5.2
- 6.3.2/6.3.8 GM_Boundary
- 6.3.4/6.3.9 GM_PrimitiveBoundary
- 6.3.5/6.3.10 GM_CurveBoundary
- 6.3.10/6.3.1 GM_Primitive – specializations B.2.5.5.1, B.2.5.5.2, B.2.5.6.1, B.2.5.6.2, B.2.5.6.5
- 6.3.11/6.3.2 GM_Point
- 6.3.13/6.3.3 GM_OrientablePrimitive
- 6.3.14/6.3.4 GM_OrientableCurve – specialization B.2.5.5.3

- 6.3.16/6.3.6 GM_Curve – specialization B.2.5.5.3
- 6.4.1/6.4.2 DirectPosition – specialization B.2.5.6.4
- 6.4.3/6.4.5 GM_Envelope
- 6.4.5/6.4.3 GM_Position – specialization B.2.5.5.4
- 6.4.6/6.4.4 GM_PointArray
- 6.4.8/6.4.7 GM_CurveInterpolation – specialization B.2.5.5.5
- 6.4.9/6.4.6 GM_CurveSegment – specializations B.2.5.5.6, B.2.5.6.3
- 6.4.10/6.4.8 GM_LineString
- 6.4.12/6.4.9 GM_GeodesicString
- 6.4.14/6.4.10 GM_ArcString
- 6.6.2/6.4.15 GM_Complex – specialization B.2.5.6.7
- 7.2.2/7.1.1 TP_Object
- 7.3.2/7.1.10 TP_Boundary
- 7.3.4/7.1.11 TP_PrimitiveBoundary
- 7.3.5/7.1.12 TP_EdgeBoundary
- 7.3.10/7.1.2 TP_Primitive – specialization B.2.5.6.5
- 7.3.11/7.1.3 TP_DirectedTopo
- 7.3.12/7.1.4 TP_Node – specialization B.2.5.6.6
- 7.3.13/7.1.5 TP_DirectedNode
- 7.3.14/7.1.6 TP_Edge – specialization B.2.5.5.7
- 7.3.15/7.1.7 TP_DirectedEdge
- 7.4.2/7.2.1 TP_Complex – specialization B.2.5.6.7

B.2.5.5 Class Specialization

B.2.5.5.1 GM_Object, GM_Primitive

The type GM_Object operation representativePoint is changed to an optional derived type GM_Primitive attribute.

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

B.2.5.5.2 GM_Object, GM_Primitive

The type GM_Object operation envelope is changed to an optional derived type GM_Primitive attribute.

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

B.2.5.5.3 GM_OrientableCurve, GM_Curve

The type GM_OrientableCurve operation boundary is changed to a derived type GM_Curve composition association to a GM_CurveBoundary object.

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

B.2.5.5.4 GM_Position

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])
```

B.2.5.5.5 GM_CurveInterpolation

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

```
GM_CurveInterpolation::

    linear

    geodesic

    circularArc3Points
```

B.2.5.5.6 GM_CurveSegment

The multiplicity of all three GM_CurveSegment::numDerivatives attributes are restricted 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])
```

B.2.5.5.7 TP_Edge

The type TP_Edge operation boundary is changed to a composition association to a TP_EdgeBoundary object.

```
TP_Edge::boundary[1] : TP_EdgeBoundary
```

B.2.5.6 Association Specialization

B.2.5.6.1 InteriorTo

The multiplicity of the association roles “coincidentSubelement” and “superElement” of GM_Primitive is restricted from [0..n] to [0].

```
GM_Primitive::coincidentSubelement[0]                                    :
Reference<GM_Primitive>                                          ([0..n])

GM_Primitive::superElement[0]     :     Reference<GM_Primitive>
                                                                 ([0..n])
```

B.2.5.6.2 Complex

The multiplicity of the association role “complex” of GM_Primitive is restricted from [0..n] to [1..n] and a GM_Primitive has exactly one maximal complex.

```
GM_Primitive::complex[1..n] : Reference<GM_Complex> ([0..n])

context GM_Primitive inv:

     self.maximalComplex()->size = 1
```

B.2.5.6.3 Segmentation

The multiplicity of the association role “curve” of GM_CurveSegment is restricted from [0,1] to [1].

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

B.2.5.6.4 Coordinate Reference System

The multiplicity of the association Coordinate Reference System of DirectPosition is restricted from [0,1] to [0].

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

B.2.5.6.5 Realization

The multiplicity of the association role “geometry” of TP_Primitive is restricted from [0,1] to [1]. The multiplicity of the association role “topology” of GM_Primitive is restructed from [0..n] to [1].

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

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

NOTE In this profile, a corresponding geometric primitive realizes each topological primitive.

B.2.5.6.6 CoBoundary

The type of the association role “spoke” of TP_Node is specialized from a Set to a Circular Sequence.

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

B.2.5.6.7 Realization

The multiplicity of the association role “geometry” of TP_Complex is restricted from [0,1] to [1]. The multiplicity of the association role “topology” of GM_Complex is restructed from [0,1] to [1].

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

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

NOTE In this profile, a corresponding maximal geometric primitive realizes the maximal topological complex.

B.2.6 Topological 2D Surface Complex Profile

B.2.6.1 Introduction

This profile consists of a collection of 0D, 1D, and 2D topological primitives that are capable of being contained within a single maximal topological complex. A corresponding geometric primitive realizes each of the individual topological primitives, and a corresponding geometric complex realizes each topological complex. All boundary and co-boundary relationships are explicitly represented. Thus, the collection of primitives makes up a fully connected composite surface.

B.2.6.2 Class Diagrams

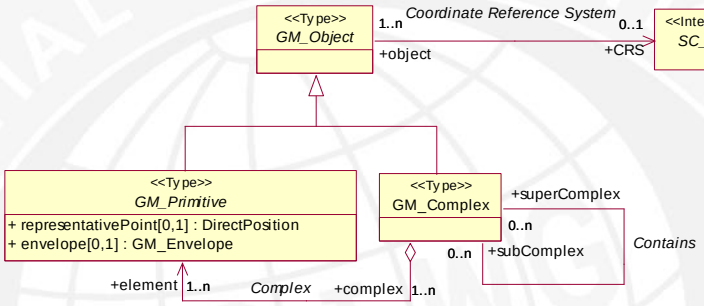


Figure B.2.6.1 – Geometry Object

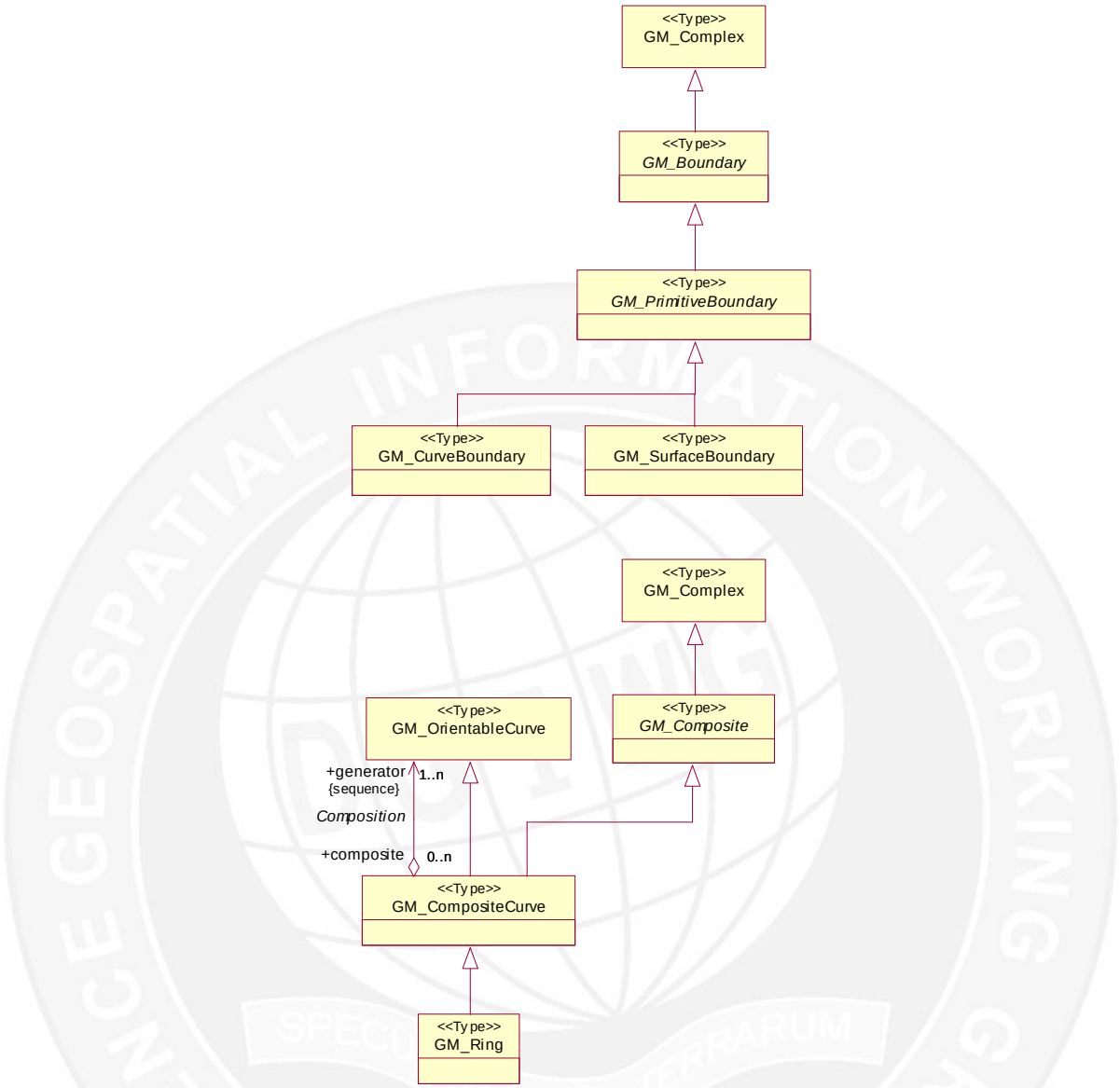


Figure B.2.6.2 – Geometry Boundary

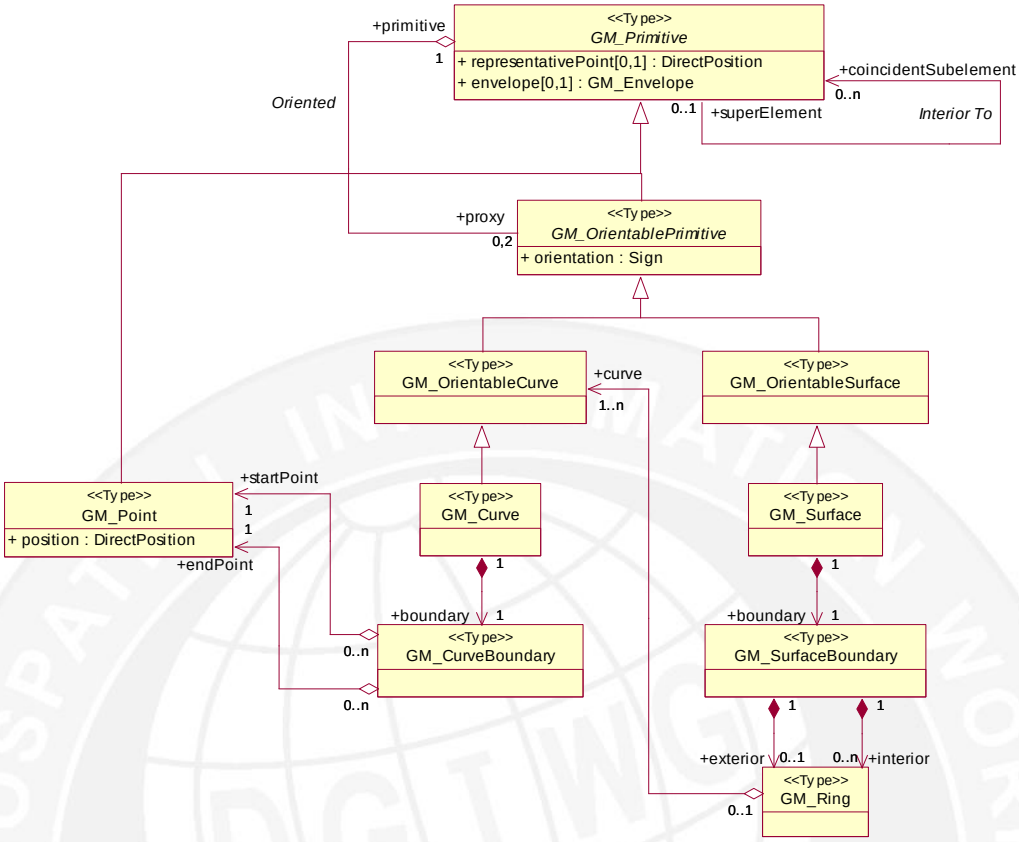


Figure B.2.6.3 – Geometry Primitive

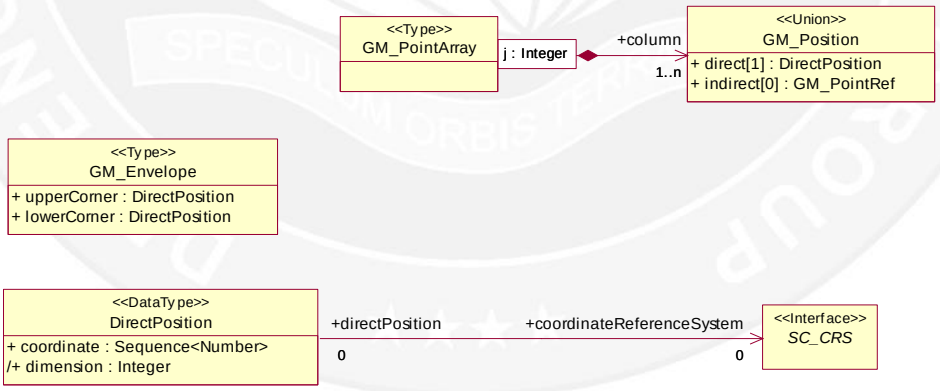


Figure B.2.6.4 – Coordinate Package

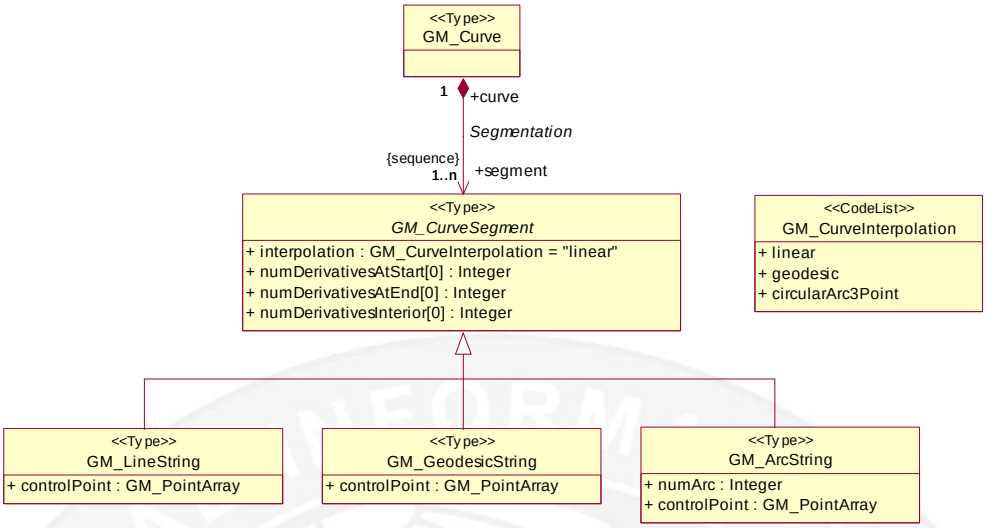


Figure B.2.6.5 – Curve Segment

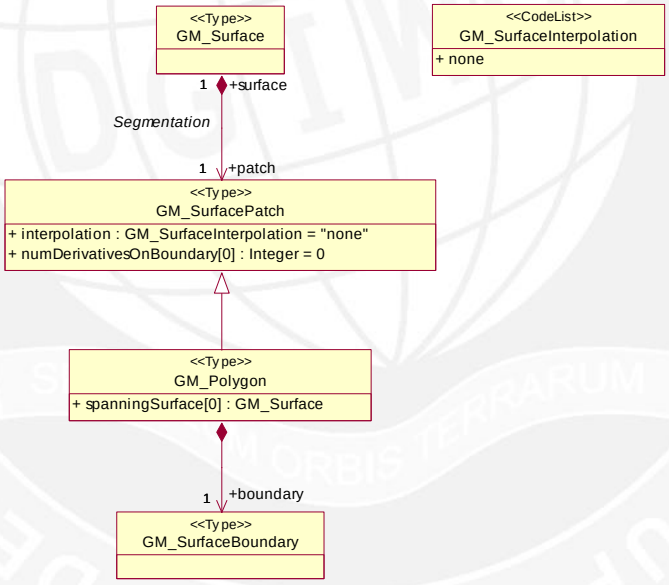


Figure B.2.6.6 – Surface Patch

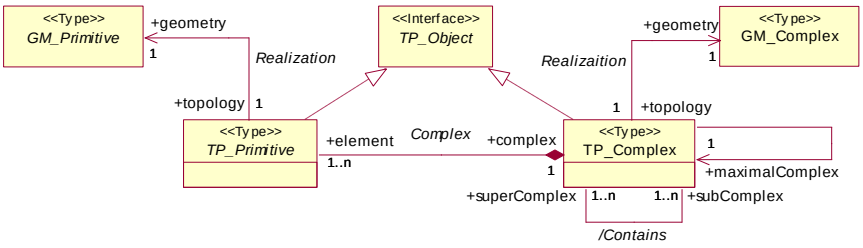


Figure B.2.6.7 – Topology Object

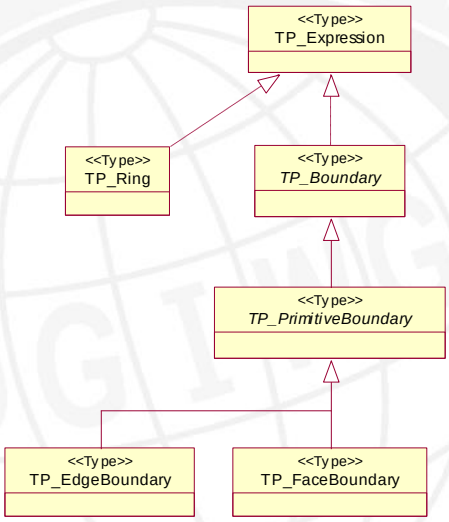


Figure B.2.6.8 – Topology Boundary

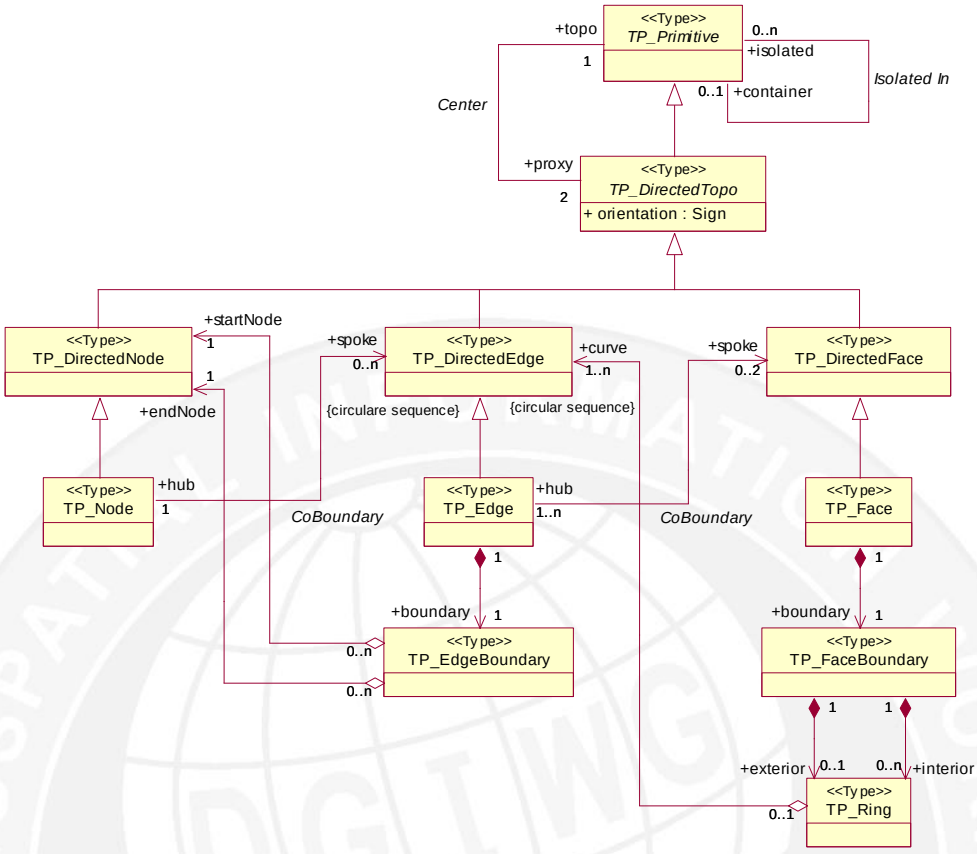


Figure B.2.6.9 – Topology Primitive

B.2.6.3 Omitted Constructs

Classes from the parent profile are omitted unless explicitly included. Associations between a class and an omitted class are also omitted.

B.2.6.4 Included Constructs

The following is a list of all the included classes preceded by their ISO 19107 section number and their DGIWG Profile 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.

- 6.2.2/6.2.1 GM_Object – specializations B.2.6.5.1, B.2.6.5.2
- 6.3.2/6.3.8 GM_Boundary
- 6.3.4/6.3.9 GM_PrimitiveBoundary
- 6.3.5/6.3.10 GM_CurveBoundary
- 6.3.6/6.3.12 GM_Ring

- 6.3.7/6.3.11 GM_SurfaceBoundary
- 6.3.10/6.3.1 GM_Primitive – specializations B.2.6.5.1, B.2.6.5.2, B.2.6.6.1, B.2.6.6.2, B.2.6.6.6
- 6.3.11/6.3.2 GM_Point
- 6.3.13/6.3.3 GM_OrientablePrimitive
- 6.3.14/6.3.4 GM_OrientableCurve – specialization B.2.6.5.3
- 6.3.15/6.3.5 GM_OrientableSurface – specialization B.2.6.5.4
- 6.3.16/6.3.6 GM_Curve – specialization B.2.6.5.3
- 6.3.17/6.3.7 GM_Surface – specializations B.2.6.5.4, B.2.6.6.4
- 6.4.1/6.4.2 DirectPosition – specialization B.2.6.6.5
- 6.4.3/6.4.5 GM_Envelope
- 6.4.5/6.4.3 GM_Position – specialization B.2.6.5.5
- 6.4.6/6.4.4 GM_PointArray
- 6.4.8/6.4.7 GM_CurveInterpolation – specialization B.2.6.5.6
- 6.4.9/6.4.6 GM_CurveSegment – specializations B.2.6.5.7, B.2.6.6.3
- 6.4.10/6.4.8 GM_LineString
- 6.4.12/6.4.9 GM_GeodesicString
- 6.4.14/6.4.10 GM_ArcString
- 6.4.32/6.4.13 GM_SurfaceInterpolation – specialization B.2.6.5.8
- 6.4.34/6.4.12 GM_SurfacePatch – specializations B.2.6.5.9, B.2.6.6.4
- 6.4.36/6.4.14 GM_Polygon – specialization B.2.6.5.10
- 6.6.2/6.4.15 GM_Complex – specialization B.2.6.6.10
- 6.6.3/6.4.16 GM_Composite
- 6.6.5/6.4.17 GM_CompositeCurve
- 7.2.2/7.1.1 TP_Object
- 7.3.2/7.1.10 TP_Boundary
- 7.3.4/7.1.11 TP_PrimitiveBoundary
- 7.3.5/7.1.12 TP_EdgeBoundary
- 7.3.6/7.1.13 TP_FaceBoundary
- 7.3.8/7.1.14 TP_Ring

- 7.3.10/7.1.2 TP_Primitive – specialization B.2.6.6.6
- 7.3.11/7.1.3 TP_DirectedTopo
- 7.3.12/7.1.4 TP_Node – specialization B.2.6.6.7
- 7.3.13/7.1.5 TP_DirectedNode
- 7.3.14/7.1.6 TP_Edge – specializations B.2.6.5.11, B.2.6.6.8
- 7.3.15/7.1.7 TP_DirectedEdge
- 7.3.16/7.1.8 TP_Face – specialization B.2.6.5.12
- 7.3.177.1.9 TP_DirectedFace
- 7.4.2/7.2.1 TP_Complex – specialization B.2.6.6.9

B.2.6.5 Class Specialization

B.2.6.5.1 GM_Object, GM_Primitive

The type GM_Object operation representativePoint is changed to an optional derived type GM_Primitive attribute.

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

B.2.6.5.2 GM_Object, GM_Primitive

The type GM_Object operation envelope is changed to an optional derived type GM_Primitive attribute.

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

B.2.6.5.3 GM_OrientableCurve, GM_Curve

The type GM_OrientableCurve operation boundary is changed to a derived type GM_Curve composition association to a GM_CurveBoundary object.

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

B.2.6.5.4 GM_OrientableSurface, GM_Surface

The type GM_OrientableSurface operation boundary is changed to a derived type GM_Surface composition association to a GM_SurfaceBoundary object.

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

B.2.6.5.5 GM_Position

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])
```

B.2.6.5.6 GM_CurveInterpolation

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

```
GM_CurveInterpolation::

      linear

      geodesic

      circularArc3Points
```

B.2.6.5.7 GM_CurveSegment

The multiplicity of all three GM_CurveSegment::numDerivatives attributes are restricted 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])
```

B.2.6.5.8 GM_SurfaceInterpolation

The code list GM_SurfaceInterpolation is constrained to the value “none”.

```
GM_SurfaceInterpolation::

      none
```

B.2.6.5.9 GM_SurfacePatch

The multiplicity of the GM_SurfacePatch::numDerivativesOnBoundary attributes is restricted from [0,1] to [0].

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

B.2.6.5.10 GM_Polygon

The multiplicity of the GM_Polygon::spanningSurface attributes is restricted from [0,1] to [0].

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

B.2.6.5.11 TP_Edge

The type TP_Edge operation boundary is changed to a composition association to a TP_EdgeBoundary object.

```
TP_Edge::boundary[1] : TP_EdgeBoundary
```

B.2.6.5.12 TP_Face

The type TP_Face operation boundary is changed to a composition association to a TP_FaceBoundary object.

```
TP_Face::boundary[1] : TP_FaceBoundary
```

B.2.6.6 Association Specialization

B.2.6.6.1 InteriorTo

The InteriorTo association only associates GM_Primitive with a dimensional difference greater than one. The multiplicity of the association role “superElement” of GM_Primitive is restricted from [0..n] to [0..1]. The association is only navigable from super-element to coincident sub-element.

```
GM_Primitive::superElement[0..1]    :    Reference<GM_Primitive>
                                                        ([0..n])

context GM_Primitive inv:

      coincidentSubelement->forAll( sub |

            sub.dimension() < self.dimension() – 1 )
```

B.2.6.6.2 Complex

The multiplicity of the association role “complex” of GM_Primitive is restricted from [0..n] to [1..n] and a GM_Primitive has exactly one maximal complex.

```
GM_Primitive::complex[1..n] : Reference<GM_Complex> ([0..n])

context GM_Primitive inv:

      self.maximalComplex()->size = 1
```

B.2.6.6.3 Segmentation

The multiplicity of the association role “curve” of GM_CurveSegment is restricted from [0,1] to [1].

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

B.2.6.6.4 Segmentation

The multiplicity of the association role “patch” of GM_Surface is restricted from [1..n] to [1]. The multiplicity of the association role “surface” of GM_SurfacePatch is restricted from [0,1] to [1].

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

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

B.2.6.6.5 Coordinate Reference System

The multiplicity of the association Coordinate Reference System of DirectPosition is restricted from [0,1] to [0].

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

B.2.6.6.6 Realization

The multiplicity of the association role “geometry” of TP_Primitive is restricted from [0,1] to [1]. The multiplicity of the association role “topology” of GM_Primitive is restructed from [0..n] to [1].

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

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

NOTE In this profile, a corresponding geometric primitive realizes each topological primitive.

B.2.6.6.7 CoBoundary

The type of the association role “spoke” of TP_Node is specialized from a Set to a Circular Sequence.

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

B.2.6.6.8 CoBoundary

The multiplicity of the association role “spoke” of TP_Edge is restricted from [0..n] to [0..2].

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

B.2.6.6.9 Realization

The multiplicity of the association role “geometry” of TP_Complex is restricted from [0,1] to [1]. The multiplicity of the association role “topology” of GM_Complex is restructed from [0,1] to [1].

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

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

NOTE In this profile, a corresponding maximal geometric primitive realizes the maximal topological complex.