Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: 2022-04-24

External identifier of this OGC® document: http://www.opengis.net/doc/dp/DynamicFeatures/{m.n}

Internal reference number of this OGC® document: 22-005

Category: OGC® Discussion Paper

Editor: Charles Heazel

OGC Dynamic Features Discussion Paper

Document type:	OGC® Discussion Paper
Document subtype:	
Document stage:	Draft
Document language:	English

Table of Contents

i. Abstract

Moving Feature standards and technologies have made considerable progress. It is time to evaluate the foundations of OGC standards to determine if they are sufficient to support current and future Moving Feature standards. This discussion paper examines ISO and OGC standards against current and anticipated future needs. It also examines other standards and conventions which may be useful for enhancing the OGC/ISO foundation.>

ii. Keywords

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

ogcdoc, OGC document, <tags separated by commas>

iii. Preface

NOTE

Insert Preface Text here. Give OGC specific commentary: describe the technical content, reason for document, history of the document and precursors, and plans for future work. > Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

Organization name(s)

v. Submitters

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

Name Affiliation

Chapter 1. Scope

Moving Feature standards and technologies have made considerable progress. It is time to evaluate the foundations of OGC standards to determine if they are sufficient to support current and future Moving Feature standards. This discussion paper examines ISO and OGC standards against current and anticipated future needs. It also examines other standards and conventions which may be useful for enhancing the OGC/ISO foundation.

Chapter 2. References

As a Discussion Paper, this document contains no normative references. Citations for resources referenced in this document are listed in the Bibliography (Annex B).

Chapter 3. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Best Practice.

For the purposes of this document, the following additional terms and definitions apply.

3.1. base representation

<moving features>representation, using a local origin and local ordinate vectors, of a geometric object at a given reference time [ISO 19141:2008, definition 4.1.1]

NOTE 1 A rigid geometric object may undergo translation or rotation, but remains congruent with its base representation.

NOTE 2 The local origin and ordinate vectors establish an engineering coordinate reference system (ISO 19111), also called a local frame or a local Euclidean coordinate system.

3.2. design coordinate reference system

engineering coordinate reference system in which the base representation of a moving object is specified. [ISO 19141:2008, definition 4.1.3]

3.3. direct position

DirectPosition object data types hold the coordinates for a position within some coordinate reference system.

3.4. external coordinate reference system

the coordinate reference system in which the geometry of the curve for a trajectory is defined. Usually an earth-fixed geographic CRS. [ISO 19141:2008, derived]

3.5. feature

abstraction of real world phenomena [ISO 19101:2002, definition 4.11]

NOTE A feature may occur as a type or an instance. Feature type or feature instance shall be used when only one is meant.

3.6. foliation

one parameter set of geometries such that each point in the prism of the set is in one and only one trajectory and in one and only one leaf [ISO 19141:2008, definition 4.1.8]

3.7. geometric object

spatial object representing a geometric set [ISO 19107:2003, definition 4.47]

3.8. geometric primitive

geometric object representing a single, connected, homogeneous element of space [ISO 19107:2003, definition 4.48]

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

3.9. global coordinate system

the CRS of the trajectory of the reference point.

NOTE A global coordinate system is usually in terms of the moving frame of the curve (oriented on the local tangent) or in terms of the external CRS in which the geometry of the curve is defined. [ISO 19141:2008, derived]

3.10. GM_Point

GM_Point is the basic data type for a geometric object consisting of one and only one point. That point is represented as a DirectPosition.

3.11. instant

0-dimensional geometric primitive representing position in time [ISO 19108:2002, definition 4.1.17]

3.12. leaf

<one parameter set of geometries> geometry at a particular value of the parameter [ISO 19141:2008, definition 4.1.12]

3.13. one parameter set of geometries

function f from an interval $t \in [a, b]$ such that f(t) is a geometry and for each point $P \in f(a)$ there is a one parameter set of points (called the trajectory of P) $P(t) : [a, b] \rightarrow P(t)$ such that $P(t) \in f(t)$ [ISO 19141:2008, definition 4.1.15]

EXAMPLE A curve C with constructive parameter t is a one parameter set of points c(t).

3.14. prism

<one parameter set of geometries> set of points in the union of the geometries (or the union of the trajectories) of a one parameter set of geometries [ISO 19141:2008, definition 4.1.18]

NOTE This is a generalization of the concept of a geometric prism that is the convex hull of two congruent polygons in 3D-space. Such polyhedrons can be viewed as a foliation of congruent polygons.

3.15. temporal coordinate system

temporal reference system based on an interval scale on which distance is measured as a multiple of a single unit of time [ISO 19108:2002, definition 4.1.31]

3.16. temporal position

location relative to a temporal reference system [ISO 19108:2002, definition 4.1.34]

3.17. temporal reference system

reference system against which time is measured [ISO 19108:2002, definition 4.1.35]

3.18. trajectory

path of a moving point described by a one parameter set of points [ISO 19141:2008, definition 4.1.22]

3.19. vector

quantity having direction as well as magnitude [ISO 19123:2005, definition 4.1.43]

Chapter 4. Conventions

This sections provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

4.1. Identifiers

The normative provisions in this document are denoted by the URI

http://www.opengis.net/spec/{standard}/{m.n}

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

NOTE	A description of UML and how it is used in ISO and OGC standards will be included here.

Chapter 5. Dynamic Features

Introductory text

5.1. Geometry in 3 Dimensions

Topic: Is the OGC Geometry model sufficient to support 3D?

5.1.1. Feature Model

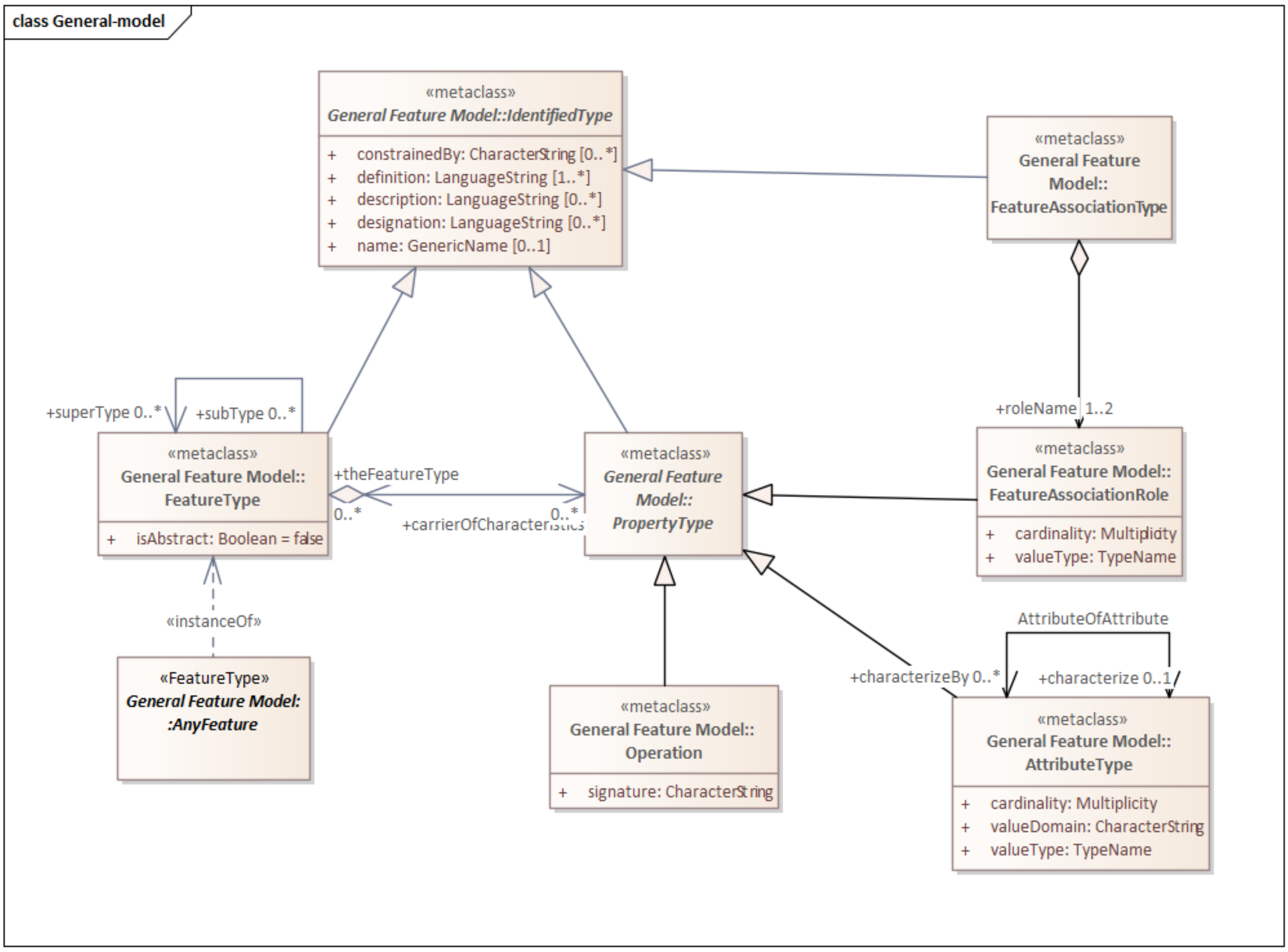
Before we can discuss geometry, we must first have a conceptual model of the universe. The OGC and ISO TC211 provide that model in ISO 19109. This standard defines the General Feature Model (GFM). The GFM defines the concept of a Feature, its components, and behaviors.

ISO TC211 defines a Feature as an "Abstraction of real world phenomena" (ISO 19101-1:2014)

A Feature, then, is a high level abstraction for anything that does or could exist in the universe.

The General Feature Model further refines the concept of a Feature. The following principles are most relevant for this discussion:

1. A Feature can be a FeatureType or an Instance of a FeatureType (AnyFeature).
2. FeatureTypes can form a taxonomy (inheritance)
3. Features possess characteristics (Properties)
4. A Property of a Feature can be an Operation, Attribute, or Association.

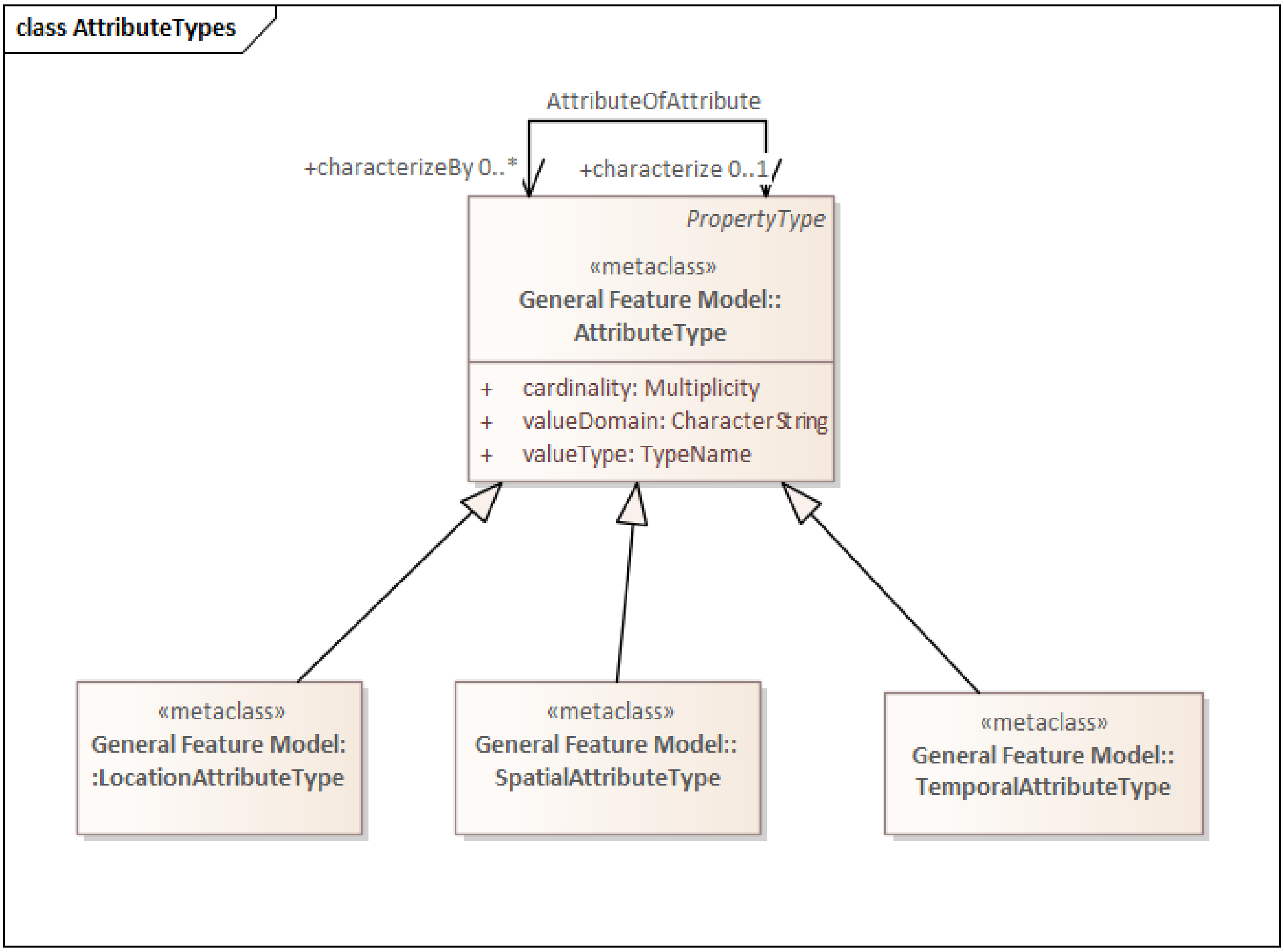


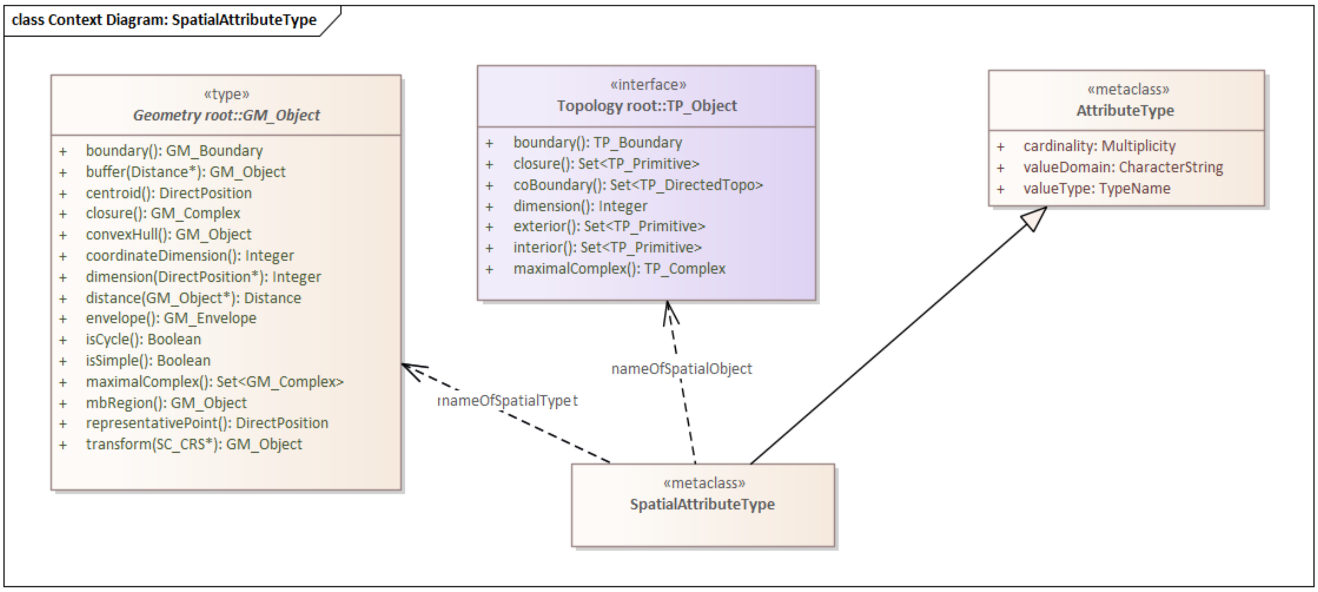
The resulting model is sufficient to describe a Feature's identity (IdentifiedType), what it is (FeatureType), what it can do (Operations), its observable characterisitics (Attributes), and any associations with other Feature instances.

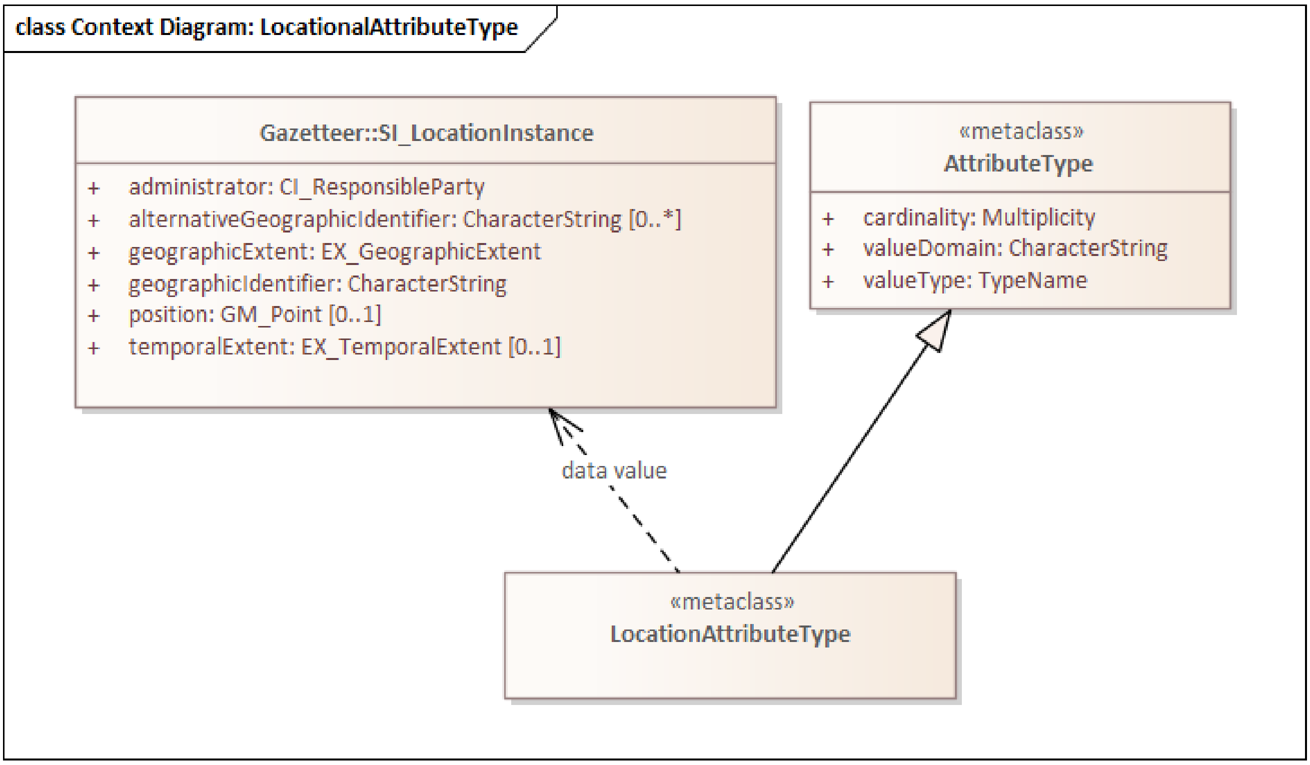
5.1.2. Feature Geometry

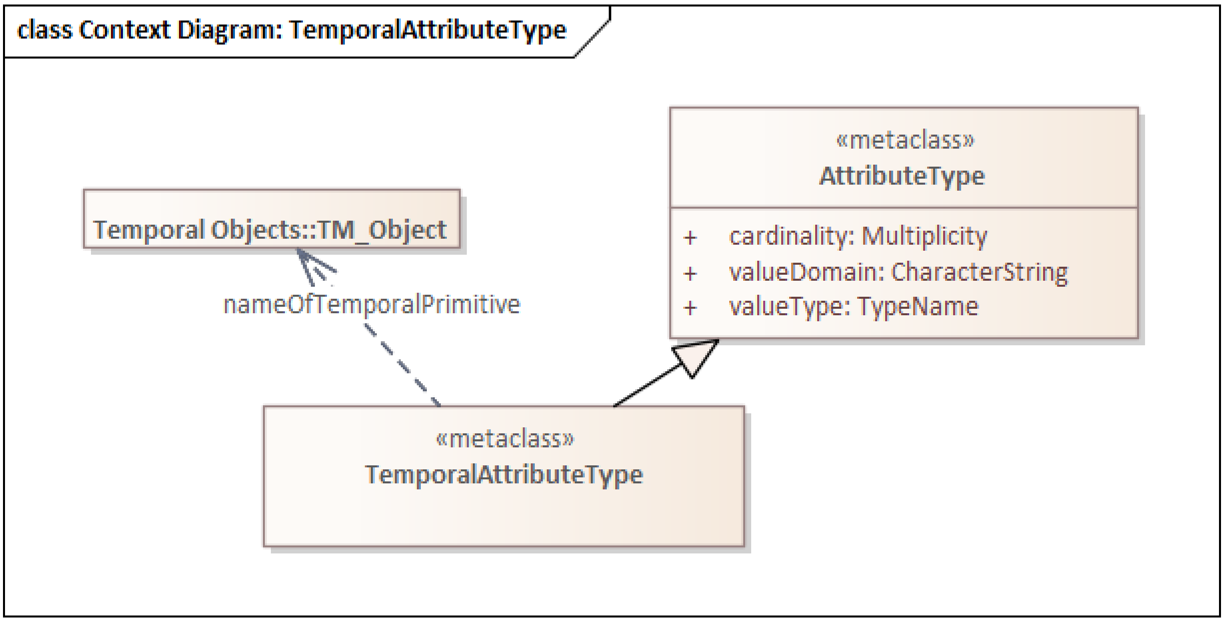
The General Feature Model treats geometry as an attribute of the Feature. In addition, it defines three types of attribute which are useful for associating geometry with a Feature in a standard manner:

- SpatialAttributeType: Geometries (GM_Object) and Topologies (TP_Object)
- LocationalAttributeType: Named locations, extents, and points.
- TemporalAttributeType: Temporal objects (TM_Object)


class AttributeTypes
AttributeOfAttribute
+characterizeBy 0..*
+characterize 0..1
PropertyType
«metaclass»
General Feature Model::
AttributeType
+ cardinality: Multiplicity
+ valueDomain: CharacterString
+ valueType: TypeName
«metaclass»
General Feature Model:
:LocationAttributeType
«metaclass»
General Feature Model::
SpatialAttributeType
«metaclass»
General Feature Model::
TemporalAttributeType


class Context Diagram: SpatialAttributeType
«type»
Geometry root::GM_Object
+ boundary(): GM_Boundary
+ buffer(Distance*): GM_Object
+ centroid(): DirectPosition
+ closure(): GM_Complex
+ convexHull(): GM_Object
+ coordinateDimension(): Integer
+ dimension(DirectPosition*): Integer
+ distance(GM_Object*): Distance
+ envelope(): GM_Envelope
+ isCycle(): Boolean
+ isSimple(): Boolean
+ maximalComplex(): Set<GM_Complex>
+ mbRegion(): GM_Object
+ representativePoint(): DirectPosition
+ transform(SC_CRS*): GM_Object
«interface»
Topology root::TP_Object
+ boundary(): TP_Boundary
+ closure(): Set<TP_Primitive>
+ coBoundary(): Set<TP_DirectedTopo>
+ dimension(): Integer
+ exterior(): Set<TP_Primitive>
+ interior(): Set<TP_Primitive>
+ maximalComplex(): TP_Complex
«metaclass»
AttributeType
+ cardinality: Multiplicity
+ valueDomain: CharacterString
+ valueType: TypeName
nameOfSpatialObject
nameOfSpatialType
«metaclass»
SpatialAttributeType





In our discussion of Dynamic Features, we must allow for Features which are moving through three dimensions and have non-trivial three dimensional shapes. We must also consider that the shape of these objects may change with time. From this we see that the movement of the Feature and the shape of the Feature are two separate properties.

A measurement of movement would capture changes in location and orientation. Movement is in the eyes of an external observer. Therefore, movement should be specified using a coordinate reference system which is external to the Feature.

The shape of a Feature is independent of its location. A rigid body has the same shape reqardless of where it is or who is observing it. It's geometry should be self-contained. This requires use of an internal coordinate reference system.

This leads up to two postulates:

Postulate 1: The Locational Attribute of a 3D Feature is a GM_Point which locates the origin of the local CRS within an external CRS.

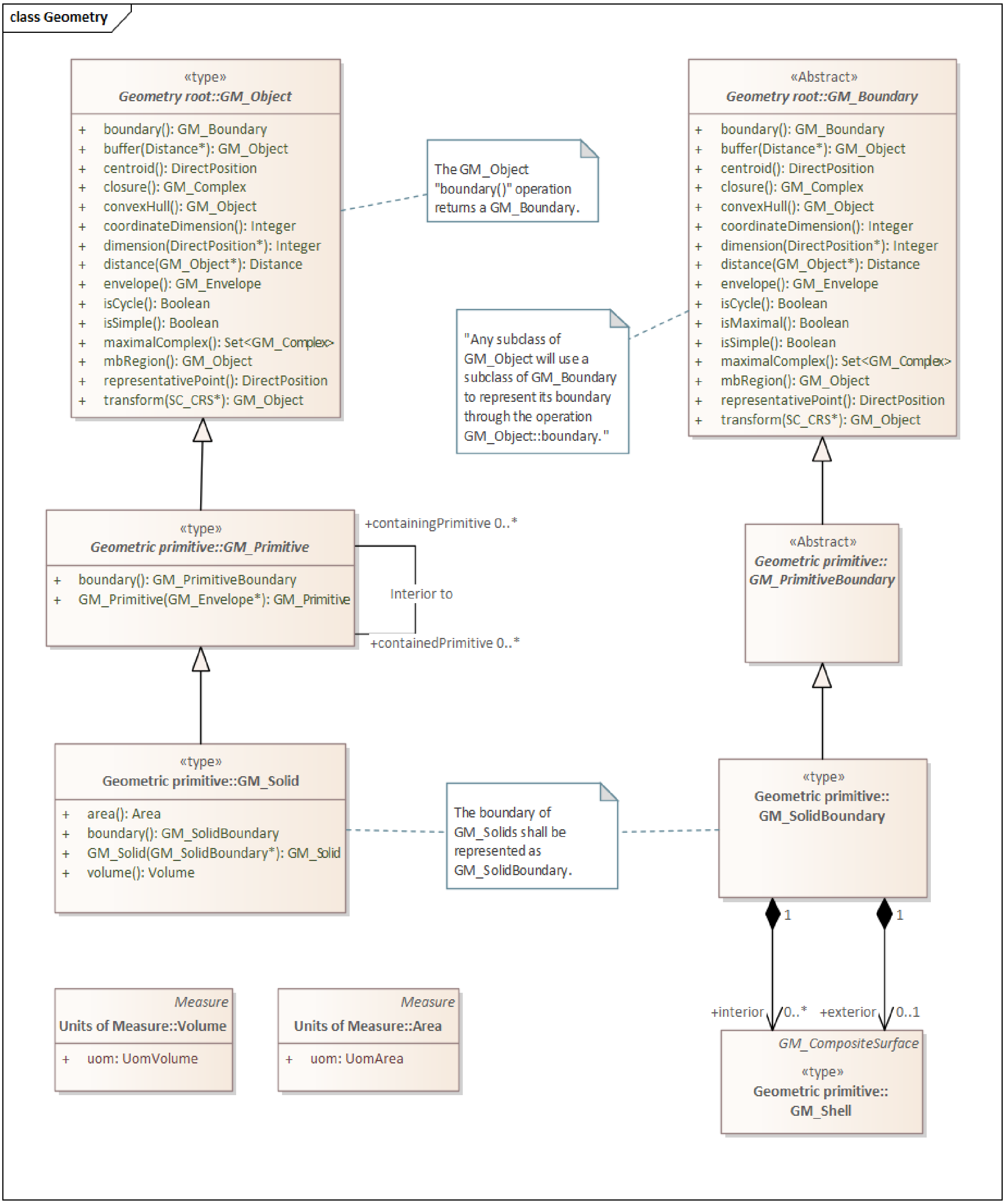
Postulate 2: The Spatial Attribute of a 3D Feature is one or more GM_Objects which define the shape of the Feature in the local coordinate refrence system.

One and two dimensional geometries are well understood. So the locational attribute poses few immediate problems. Three-dimensional shapes are another matter. Particularly if they are free from any bounding surface and capable of movement.

5.1.3. 3D Geometry

The applicable OGC/ISO standard for 3D geometries is ISO 19107:2003. While a new version was approved in 2019, it hasn't propagated through the rest of the ISO standards baseline. As a result, 19107:2003 is the most recent implemented version.

ISO 19107 defines both the GM_Object and GM_Boundary classes as root level geometry classes. The association between GM_Object and GM_Boundary is achieved through the "boundary()" operation on the GM_Object class.



Therefore, we can describe the shape of a Feature in terms of the space occupied by that Feature and the boundary which encloses that space.

Postulate 3: A 3D Feature is a Feature whos' geometry is defined by a volume enclosed by a boundary surface.

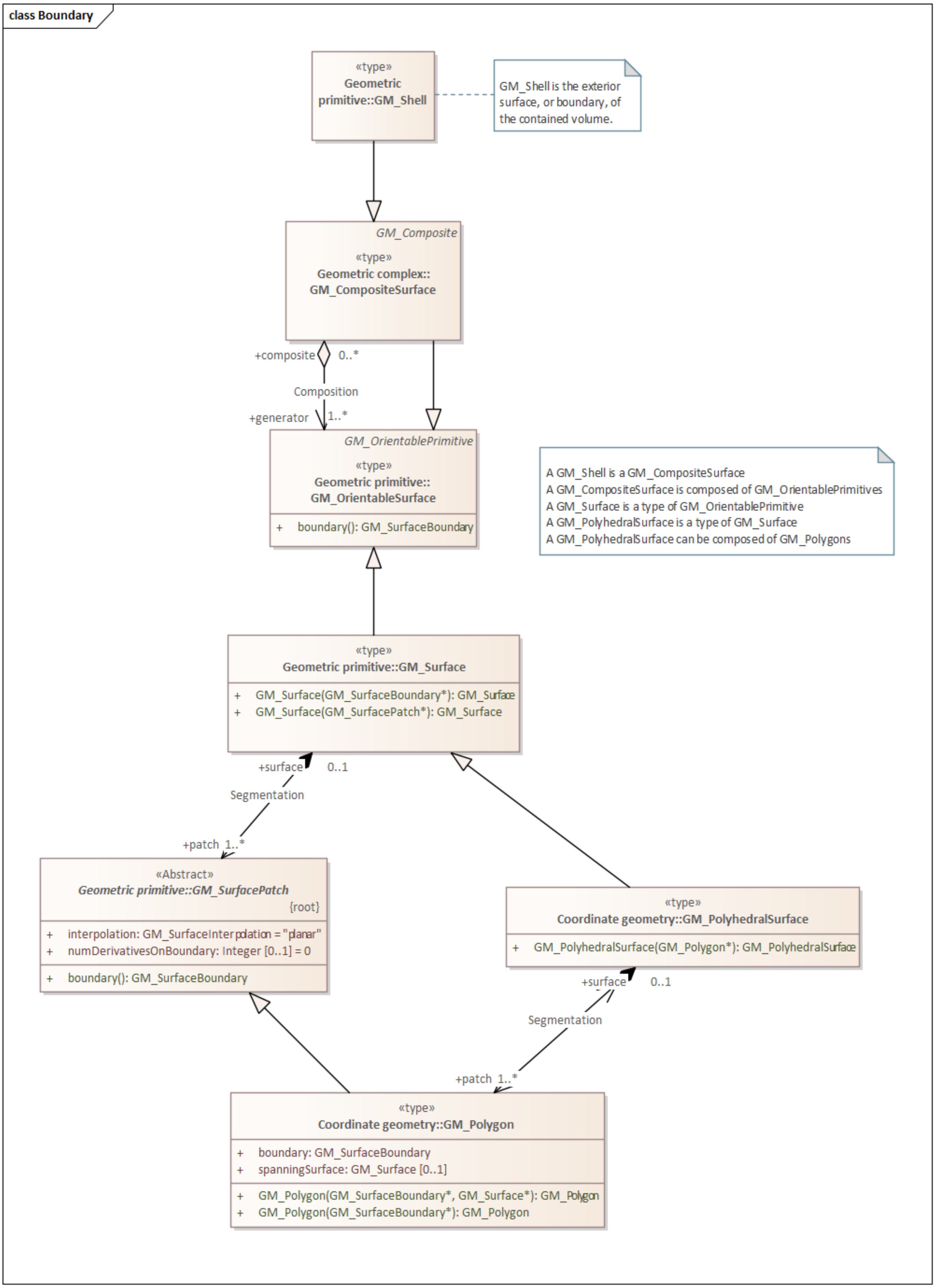
Volumetrics

GM_Object is subclassed into GM_Primitive then into GM_Solid. The “volume()” operation on GM_Solid returns the volume (defined in ISO 19103) associated with a Feature. Thus, ISO 19107 supports the concept of a 3D volume.

Real 3D objects are often not solid. So the 3D model must also support voids, or even entire 3D Features within their interior. GM_Primitive addresses this need through the “interior to” association. The two roles on this association are the containingPrimitive (the GM_Primitive which contains another GM_Primitive) and the containedPrimitive (the GM_Primitive which is contained). This association has proven its worth in 2D space so there is little doubt that it will be just as effective in 3D.

Boundaries

A 3D volume is delineated by a bounding surface. GM_Boundary is the root class for boundaries. The subclass GM_PrimitiveBoundary provides the boundary for GM_Primitives. The GM_PrimitiveBoundary subclass GM_SolidBoundary is defined as the boundary for a GM_Solid.



ISO 19107 goes even farther. A GM_SolidBoundary is composed of both interior and exterior boundaries. These boundaries are defined by the GM_Shell class. Following the class associations we see:

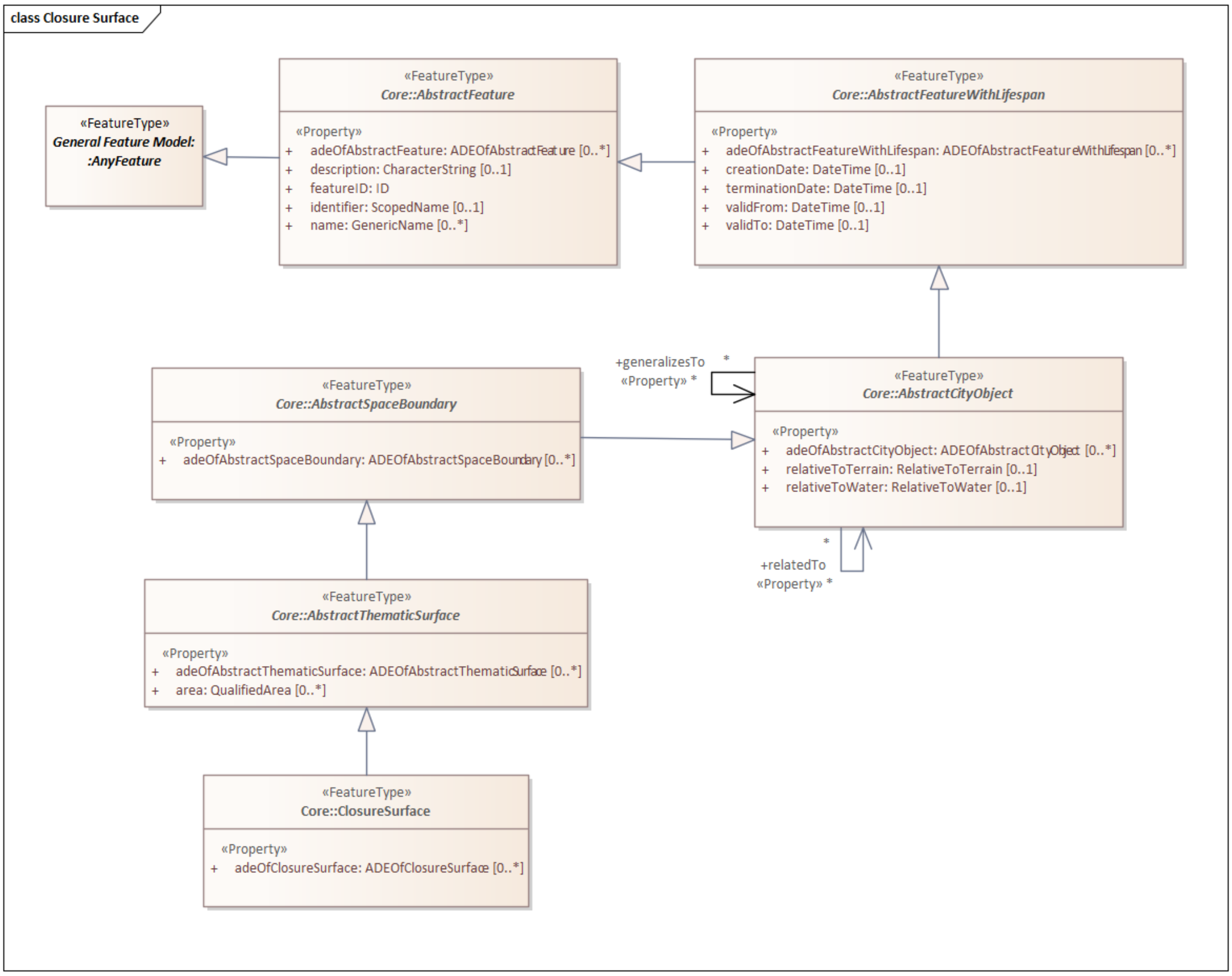
- A GM_Shell is a GM_CompositeSurface
- A GM_CompositeSurface is composed of GM_OrientablePrimitives
- A GM_Surface is a type of GM_OrientablePrimitive
- A GM_PolyhedralSurface is a type of GM_Surface
- A GM_PolyhedralSurface can be composed of GM_Polygons

A surface constructed of polygons is an example of Boundary Representation (B-Rep) of a surface. This approach is fundamental to rendering 3D computer graphics. (ref Adam Powers 1981)

Closure Surfaces

Some structures, such as a tunnel or overpass, pose difficulties for this geometry model. The boundary surface can be constructed so that it continues into the interior of the structure. That would make the interior of a tunnel external to the tunnel object. This is not always a desireable result. CityGML provides the concept of a "Closure Surface".

A Closure Surface is a surface which is a logical part of the object but does not correspond to a physical part of the object. For example, the entrance to a tunnel can have a closure surface. This surface allows you to treat the tunnel as a single three-dimension entity, even though there is a hole in the bounding surface.



As implemented in CityGML 3.0, the ClosureSurface class has quite an ancestory. We may want to generalize this concept for use outside of CityGML. However, the capabilties provided by the ancestor classes do provide value and may be worth incorporating into a general 3D model.

B-Rep

The polyhedral surfaces which bound volumetric shapes are similar to the Boundary Representation (B-Rep) approach used in CAD and computer graphics. B-Rep defines a 3-dimensional surface which serves as the interface between the interior of the volumetric shape and the exterior. This surface is usually defined by a collection of shape elements (polygons) which together form a closed surface.

https://en.wikipedia.org/wiki/Boundary_representation

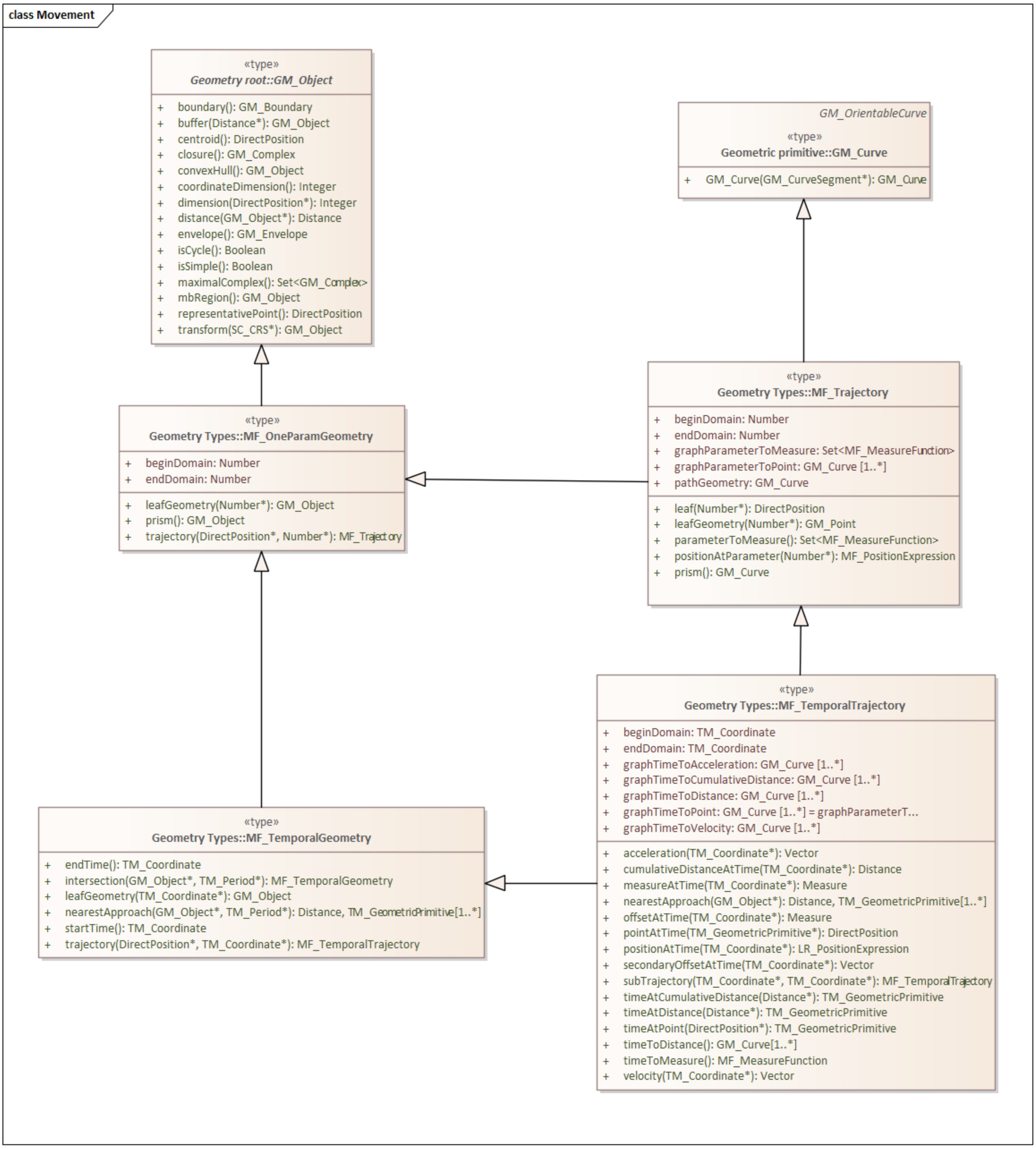
Point Clouds

"In addition to the spatial representations defined in the Core module, the geometry of physical spaces and of thematic surfaces can now also be provided by 3D point clouds using MultiPoint geometry. This allows, for example, spatially representing the building hull, a room within a building or a single wall surface just by a point cloud. All thematic feature types including transportation objects, vegetation, city furniture, etc. can also be spatially represented by point clouds. In this way, the ClearanceSpace of a road or railway could, for instance, be modelled directly from the result of a mobile laser scanning campaign. Point clouds can either be included in a CityGML dataset or just reference an external file of some common types such as LAS or LAZ."

5.2. Moving Features

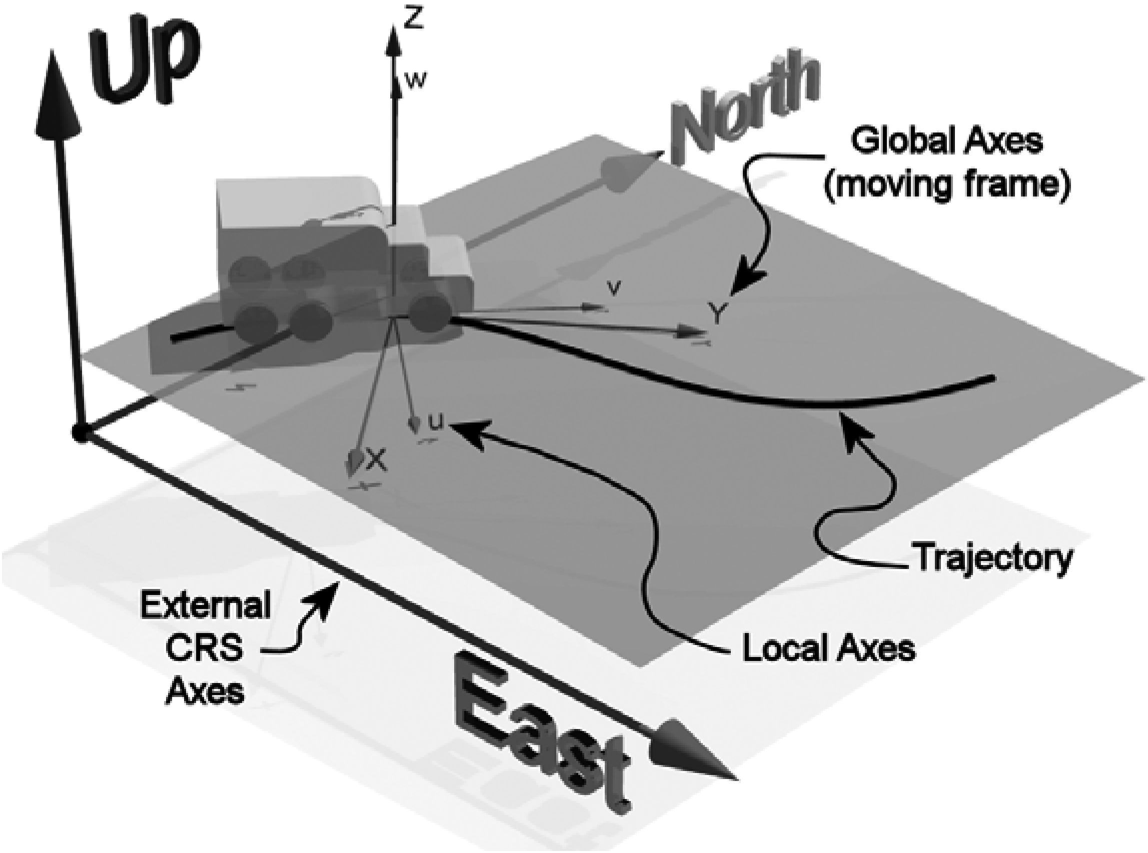
Topic: Is the OGC geometry model sufficient to support 3D Moving Features?

The ISO Standard for Moving Features is ISO 19141:2008. This standard extends the geometry model from ISO 19107 and, by association, the Feature Model from ISO 19109.



A high level view of ISO 19141 is provided in Figure --. The classes identified in this figure are described below. But first, a discussion of coordinate reference systems (CRS) is in-order.

5.2.1. Coordinate systems



Moving Features deal with three spatial coordinate systems as well at one temporal coordinate system. The spatial coordinate systems are referred to as the External, Global, and Local CRS.

When dealing with Moving Features, we frequently need to convert coordinates between the three spatial CRS. The GM_Object class provides us with the transform() operation which can be used for this purpose.

External CRS

The External coordinate system is the coordinate system within which the Moving Feature exists. Typically this is an Earth-centric geographic CRS. For example, WGS 84.

Global CRS

The Global coordinate system is a moving CRS which describes the area external to the Feature at a specific time. Typically the origin is the location of the Feature on the trajectory and the principal axis is the tangent to the trajectory at that point.

Using an aircraft for an example, the origin would be the center of mass, positive X is the magnetic heading, the Z axis is perpendicular to the tangential plane, and the Y axis is on the tangential plane and completes a right-handed coordinate system.

Local CRS

The local coordinate system is internal to the Feature. This is usually a cartisian coordinate system with the origin at a prominant point in the Feature such as the center of mass. The origin of the local coordinate system should also be the point where the location and orientation of the Feature is measured.

Temporal Reference Systems

ISO 19108:2006 Geographic Schema - Temporal Schema is the ISO standard for Temporal Reference Systems. In particular, the TM_ReferenceSystem class.

TM_ReferenceSystem has two attributes; `domainOfValidity` and `name`. The `name` attribute is an identifier for this temporal reference system. The `domainOfValidity` specifies the spatial extent over which this TRS is applicable.

TM_ReferenceSystem is specialized through a number of subclasses. The two most relevant to this paper are TM_CoordinateSystem and TM_Clock.

TM_CoordinateSystem is "A system for measuring time on a continuous interval scale using a single standard time interval". The standard time interval is provided through the `interval` attribute. In addition, the `origin` attribute provides a temporal "datum" from which time is measured. Since time is a one-dimensional quantity, the origin and interval are sufficient to define a basic Temporal Coordinate Reference System.

TM_Clock is "A system for measuring temporal position within a day". It has an optional `dateBasis` association with a calendar (TM_Calendar).

This combination of classes allows us support high precision local-clock TRS as well as full date-time TRS.

5.2.2. Coordinate Representation

The coordinates used to define a 3D moving geometry (MG) face requirements specific to their use. These requirements are derived from two characteristics of moving geometries. Unlike static spatial geometries, time and location in moving geometries are tightly coupled. They must act as a single, four dimension location. In addition, there will be a large number of coordinate measurements. This is a result of the need to accuratly track movement over time.

1. A MF coordinate must represent a discrete location in space and time.
2. A MF coordinate must include values for all three spatial axis (X,Y,Z) as well as the temporal axis (t).
3. A MF coordinate must be concise.

Of the Moving Feature encoding standards, the JSON encoding comes closest to meeting these requirements. Its major shortfall is the need for conformance with GeoJSON. Since GeoJSON assumes a terrestrial spatial geometry, spatial and temporal coordinates must be encoded separately.

- A LinearTrajectory object SHALL be a GeoJSON Feature object that has two MANDATORY members of "geometry" and "properties".

The spatial locations are captured using the GeoJSON "geometry" property. This property is restricted as follows:

- The value of the "geometry" member SHALL be a LineString Geometry object, having "type" = "LineString".
- The number of elements in the array of the "coordinates" value in the Geometry object SHALL more than two positions.

So the spatial geometry is a linestring of more than two points.

GeoJSON does no support temporal coordinates directly. So the "properties" property is adapted for this purpose. Since "properties" is not limited to temporal coordinates, these requirements are more complex.

- The value of the "properties" member SHALL be a GeoJSON object that has at least a member with the named "datetimes".
- The value of the "datetimes" member is a JSON array.
- Each element in the array of the "datetimes" value SHALL be an instant object.
- An instant object SHALL be only a JSON string to represent a timestamp encoded by the IETF RFC 3339 format using `Z` or the numeric value of milliseconds since midnight (00:00 a.m.) on January 1, 1970, in UTC.
- The array of the `"datetimes"` value SHALL be a monotonic increasing sequence.
- There SHALL be no instant object that has the same value as any other element.

The consequence of these requirements is that the "properties" property can carry the temporal equivalent to a line string. There is one final requirement.

- The number of elements in both arrays of the "coordinates" value and the "datetimes" value SHALL be equal.

So there is a one-to-one correspondance between the temporal measurements in the "properties" property and the spatial measurements in the "geometry" property.

An example of this encoding is provided ---

```
{
    "type": "Feature",
    "id": "A",
    "geometry": {
        "type": "LineString",
        "coordinates": [[11.0,2.0,50.0], [12.0,3.0,52.0], [10.0,3.0,56.0]]
    },
    "properties": {
        "datetimes": ["2012-01-17T12:33:51Z", "2012-01-17T12:33:56Z", "2012-01-
17T12:34:00Z"],
        "state": ["walking", "walking"],
        "typecode": [1, 2]
    }
},
```

5.2.3. Geometry

MF_OneParameterGeometry

We start our discussion with the class MF_OneParameterGeometry. MF_OneParameterGeometry is a subclass of GM_Object. So moving features have the 3D geometric properties of any other GM_Object. What is different is that this geometry can change as a function of a parameter.

A one parameter set of geometries is defined as "a function f from an interval t Î [a, b] such that f(t) is a geometry and for each point P Î f(a) there is a one parameter set of points (called the trajectory of P) P(t) : [a, b] ® P(t) such that P(t) Î f(t). A leaf of a one parameter set of geometries is the geometry f(t) at a particular value of the parameter".

A one parameter geometry instance includes a "leafgeometry()" operation. This operation takes the parameter (t) as input and returns the leaf P(t) for that parameter as a GM_Object.

MF_TemporalGeometry

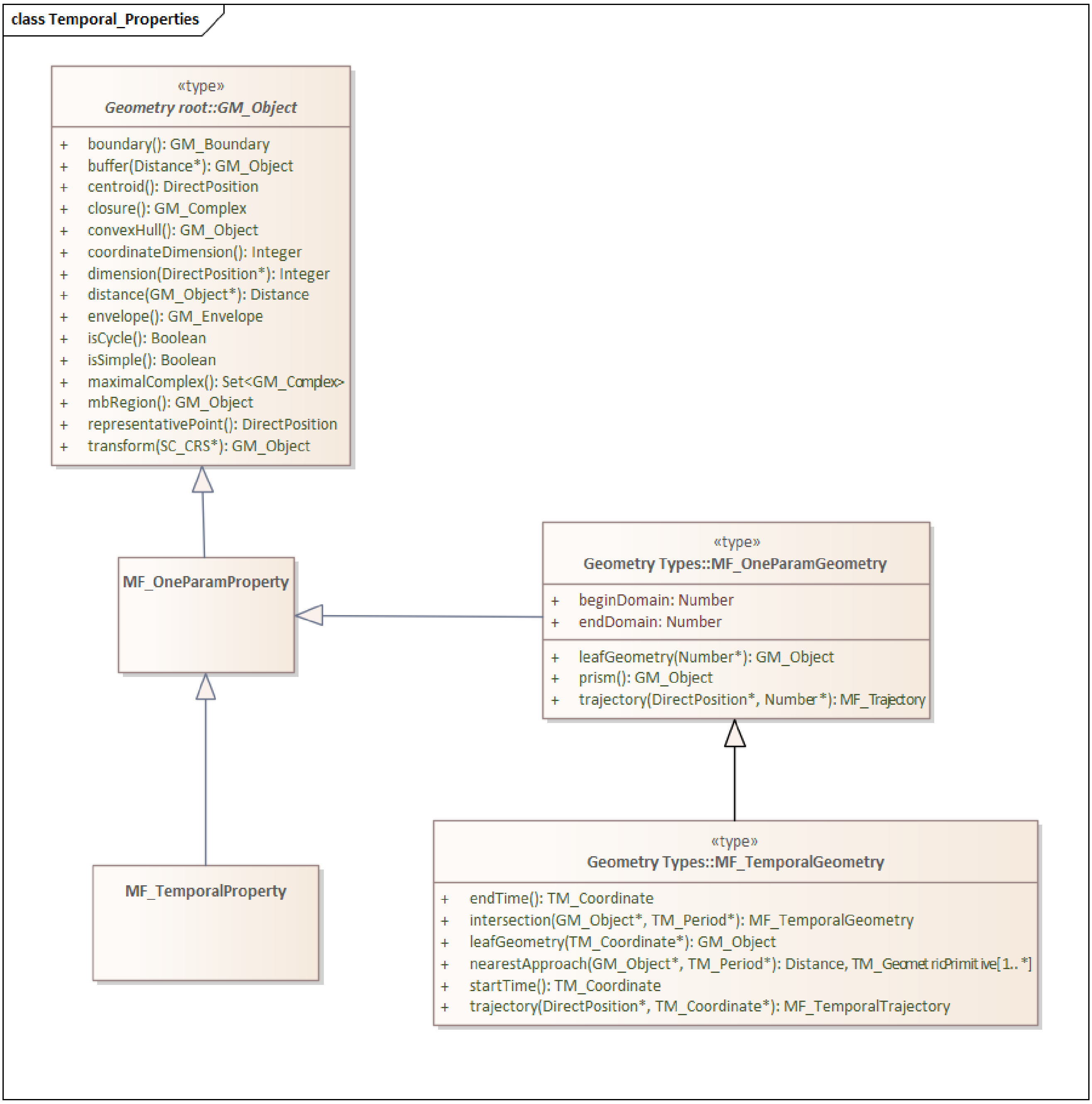
An MF_TemporalGeometry is a MF_OneParameterGeometry where the parameter is Time expressed as a TM_Coordinate. TM_Coordinate is specified in ISO 19108; it expresses time as a multiple of a single unit of measure such as year, day, or second. The "leafgeometry()" operation of an instance of MF_TemporalGeometry would take a TM_Coordinate in as input and return a GM_Object instance representing the geometry of the Feature at the specified point in time.

Temporal Properties

The JSON encoding of the OGC Moving Features standard introduces the concept of temporal properties.

"A TemporalProperties object is a JSON array of ParametricValues objects that groups a collection of dynamic non-spatial attributes and its parametric values with time."

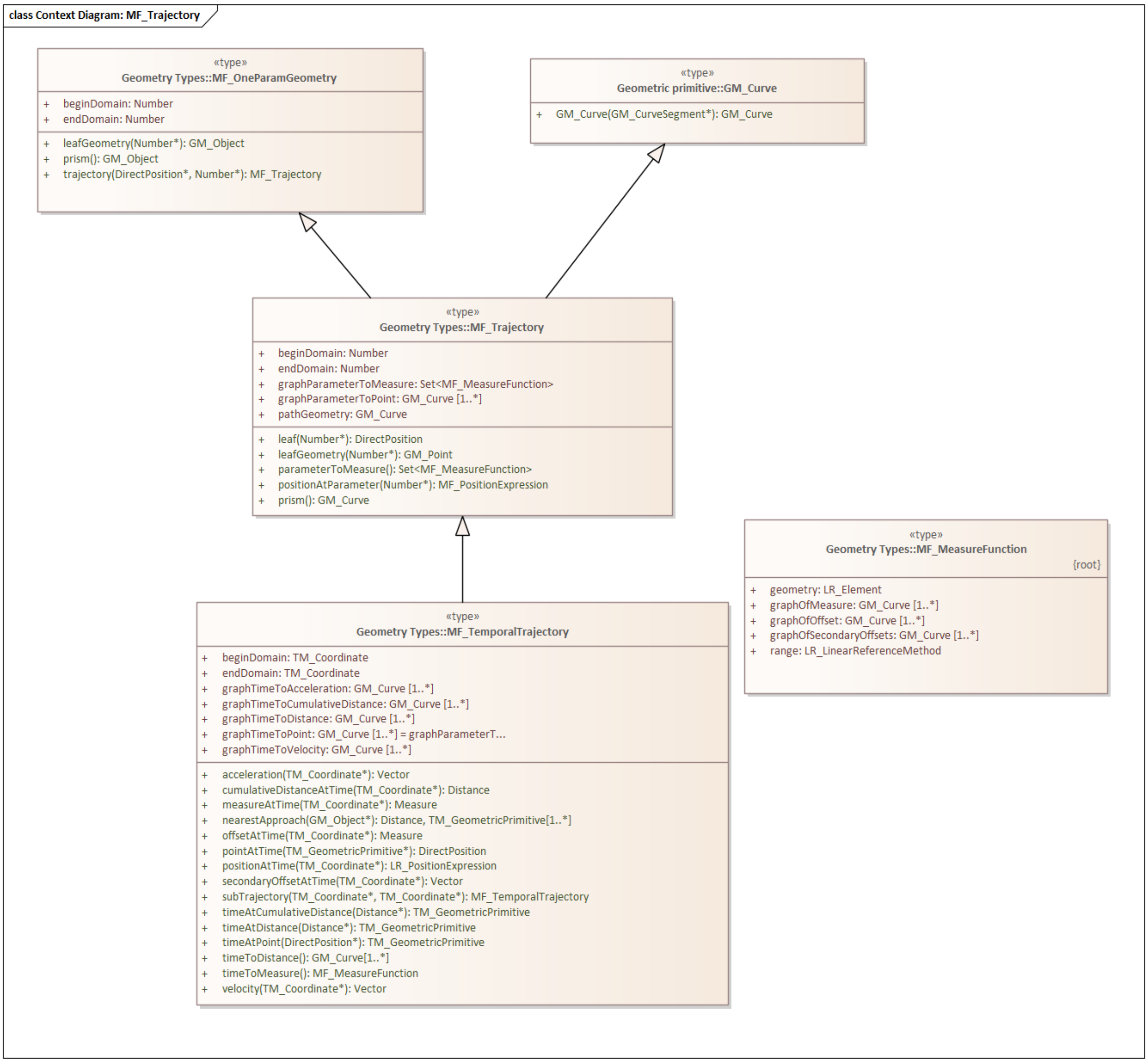
Logically TemporalProperties should be a subclass of OneParamProperties. Since Geometry is a property, then MF_TemporalGeometry should be a subclass of TemporalProperties. Which gives us the following UML.



Temporal properties are particularly useful for capturing state change. For example, the fuel load of an aircraft will change over time. The leafproperty() operation on a temporal fuel_load object would return the amount of fuel onboard at the specified time.

5.2.4. Location

ISO 19141 represents the location of a Moving Feature using two classes; MF_Trajectory and MF_TemporalTrajectory.



A MF_Trajectory is a curve (GM_Curve). It represents every postion that the Feature has occupied during it’s journey. It does not necessarily represent the time when each location was reached.

MF_TemporalTrajectory makes the MF_Trajectory a MF_TemporalGeometry. It represents location along the trajectory as a function of time. So each location is fully defined in both space and time.

A Temporal Trajectory has two operations of particular interest; leaf() and leafgeometry(). The input parameter for these operations is always time (TM_Coordinate).

The leaf() operation returns the spatial location (Direct_Position) that the Moving Feature passes at the time (TM_Coordinate) specified by the input parameter. This is a point on the trajectory GM_Curve geometry. It also serves as the origin of the Global CRS at that location on the trajectory.

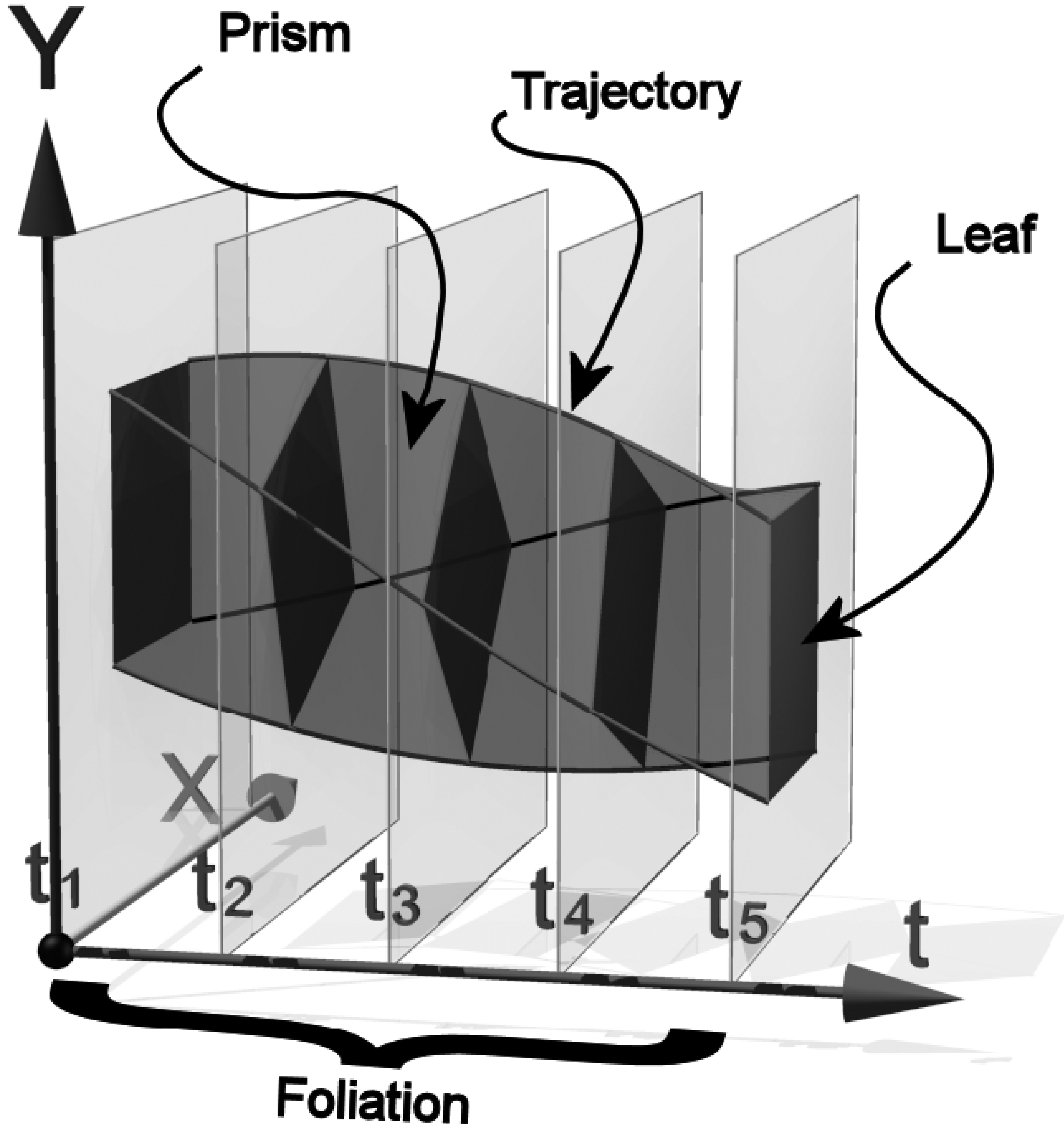
The LeafGeometry() operation returns the spatial geometry (GM_Point) that is Moving Feature passes at the time (TM_Coordinate) specified by the input parameter. This is the geometry of the Moving Feature expressed in the Local CRS. Since Trajectories only convey location, only GM_Point geometries are supported.

5.2.5. Orientation

MF_PrismGeometry

If an application focuses on only the linear movement (i.e., the spatiotemporal line string) of moving points based on World Geodetic System 1984, with longitude and latitude units of decimal degrees, and the ISO 8601 standard for representation of dates and times using the Gregorian calendar, the application can share the trajectory data by using **only** IETF GeoJSON, called **MF-JSON Trajectory**. For other cases, **MF-JSON Prism** can be used for expressing more complex movements of moving features. **MF-JSON Prism** is a GeoJSON-like format reserving new members of JSON objects (`"temporalGeometry," "temporalProperties," "crs," "trs," "time," and others) as "foreign members" to represent spatiotemporal geometries, variations of measure, coordinate reference systems, and the particular period of moving features in a JSON document.

A trajectory provides the location of a Moving Feature as a function of time. Prism Geometry represents the full geometry (location, orientaion, and shape) of the Feature as a function of time.



The key concepts in the Prism model are:

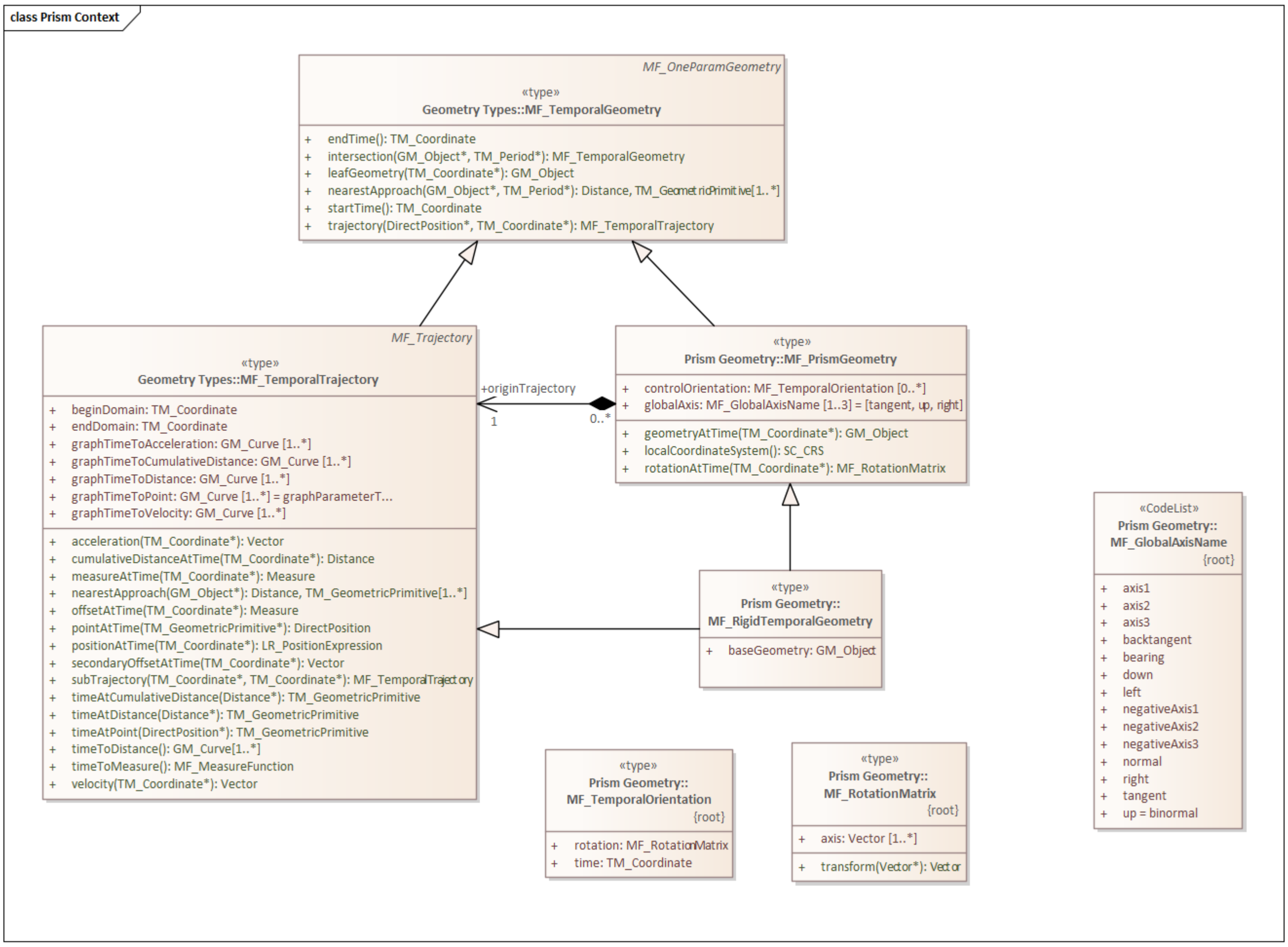
Leaf: A leaf is the geometry of the Moving Feature at time (tn).

Foliation: A collection of leaves where there is a complete and separate representation of the geometry of the Feature for each specific time (tn).

Trajectory: A curve that represents the path of a point in the geometry of the Moving Feature as it moves with respect to time (t).

Prism: the union of the geometries (or the union of the trajectories) in a foliation.

Like a Temporal Trajectory, a Prism is a subclass of MF_TemporalGeometry.



A MF_PrismGeometry class has the following characteristics.

The association role "originTrajectory" associates a Temporal Trajectory with a Prism. For any TM_Position, the associated Temporal Trajectory provides the location of the Moving Feature in the Global CRS. This point serves as the origin of the Local CRS.

The localCoordinateSystem() operation returns a SC_CRS for the design coordinate reference system in which the moving feature’s shape is defined.

The rotationAtTime() operation accepts a time in the domain of the prism geometry and returns the rotation matrix that embeds the local geometry into geographic space at a given time (TM_Coordinate). The vectors of the rotation matrix allow the feature to be aligned and scaled as appropriate to the vectors of the global coordinate reference system.

This one association and two operations provide us with the location, orientation, axis definition, and units of measure needed to define identify the local CRS and to transform geometries between the Local and Global CRS.

Finally, the geometryAtTime() operation accepts a time in the domain of the prism geometry and returns the geometry of the moving feature, as it is at a given time in the global coordinate reference system. The return type is a GM_Object so this operation is not limited to points. It is fully capable of representing a 3D surface and volume.

In short, a MF_PrismGeometry provides us with the shape, location, and orientation of a Moving Feature as a funtion of time (tn).

5.2.6. Non-rigid Bodies

ISO 19141 only addresses rigid bodies. The shape returned by a geometryAtTime() operation will always be the same. However, it leaves open the opportunity to extend the Moving Feature model to support plastic (non-rigid) objects.

The most obvious approach is to allow the geometry returned by the geometryAtTime() operation to change as a function of time. This doesn't require a change to the model. But it may require some changes to the standard.

As a correlary to this approach, the geometry itself could include MF_TemporalGeometry elements. These elements would each have their own lifespan. A history of their movement, in respect to the local CRS, over time.

5.3. Articulated Geometries

Given a suite of standards which allow you to define time-variant geometric elements, then the next step is to take a collection of those elements and assemble them into a complex object.

An articulated geometry is such an aggregation where each element has one or more but less than 6 degrees of freedom. Each element can move, but its movement is constrained by attachment to one or more additional elements.

The aggregate as a whole can also move, but that movement becomes more complex. Typically we would model movement of the whole as a trajectory of the center of mass. However, the center of mass of an articulating Feature may change as the relative position of the elements change.

Two existing standards address the problem of Articulated Geometries. The MISB Staging System was developed for articulated Motion Imagery capture systems. GeoPOSE also addresses Motion Imagery, but it is more focused on Augmented and Virtual reality.

It is not likely that either standard provides a complete solution. Their relative merits and deficiencies are discussed in the `Discussion` section as well as proposals for work to be done.

5.3.1. MISB Staging System

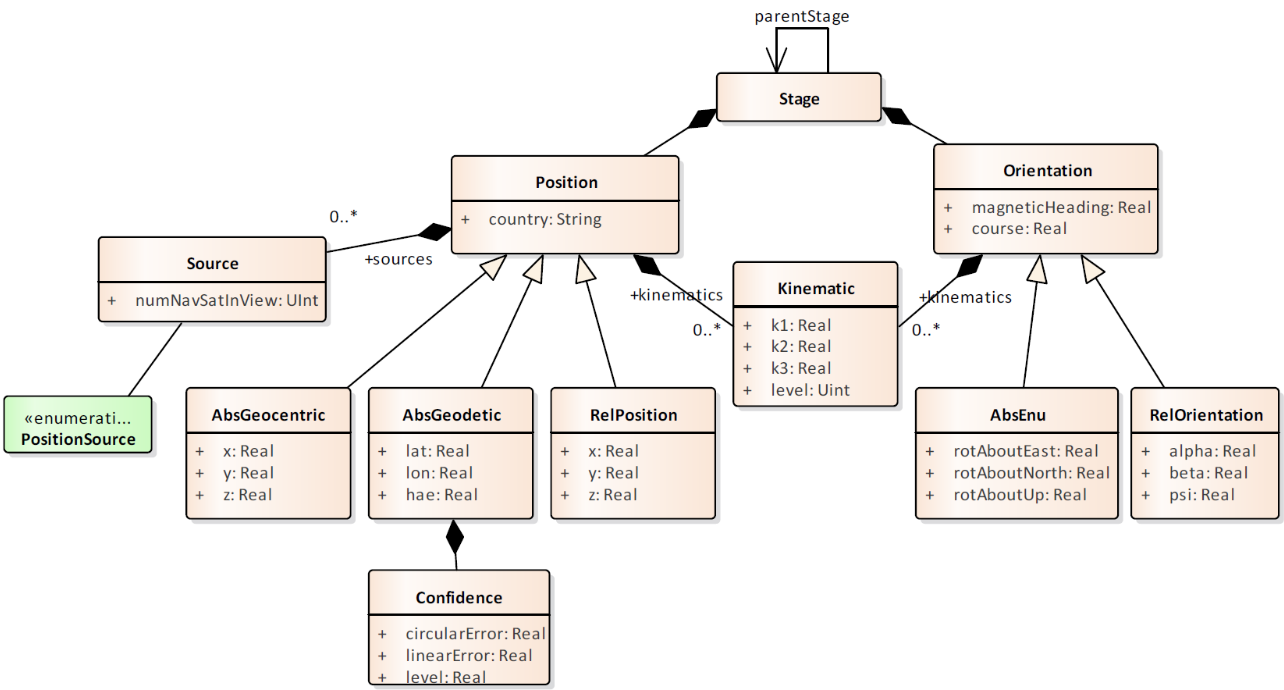
The Motion Imagery Standards Board (MISB) develops standards for motion imagery, audio, associated metadata, and their related systems. Exploitation of MISB data requires its precise positioning in space and time. Therefore, a number of MISB standards deal with the derivation of ground coordinates from sensor coordinates.

Motion Imagery systems are typically complex structures consisting of multiple components each capable of independent motion. Measurements of these motions are in a local coordinate reference system. Conversion of a sensor coordinate requires a series of transformations, starting from the detector and working back through the attached components to a core, geolocated component.

Further complicating maters, each component is capable of movement. The transformations are not fixed. They must be calculated from the relative position of each component at the time that the detection was made. This leads to a requirement that for every observation, the relative position of each component of the collecting system must be captured as well.

The MISB 1906 Motion Imagery Metadata (MIMD): Staging System standard addresses this requirement. This standard captures the locations, orientations, and kinematics (velocity, acceleration, etc.) of platforms, gimbals, sensors, sensor elements, geospatial and other points. The locations and orientations are either absolute references to a well-known frame of reference (e.g., WGS84) or relative references to other locations and orientations. Each location and orientation pairing define a “stage” that has the potential to be the frame of reference for another location and orientation. Linking stages together forms the Staging System. The Staging System then defines an ability to describe, in metadata, the physical make-up and configuration of a system and the time varying physical relationships of the system and its sub-system components.

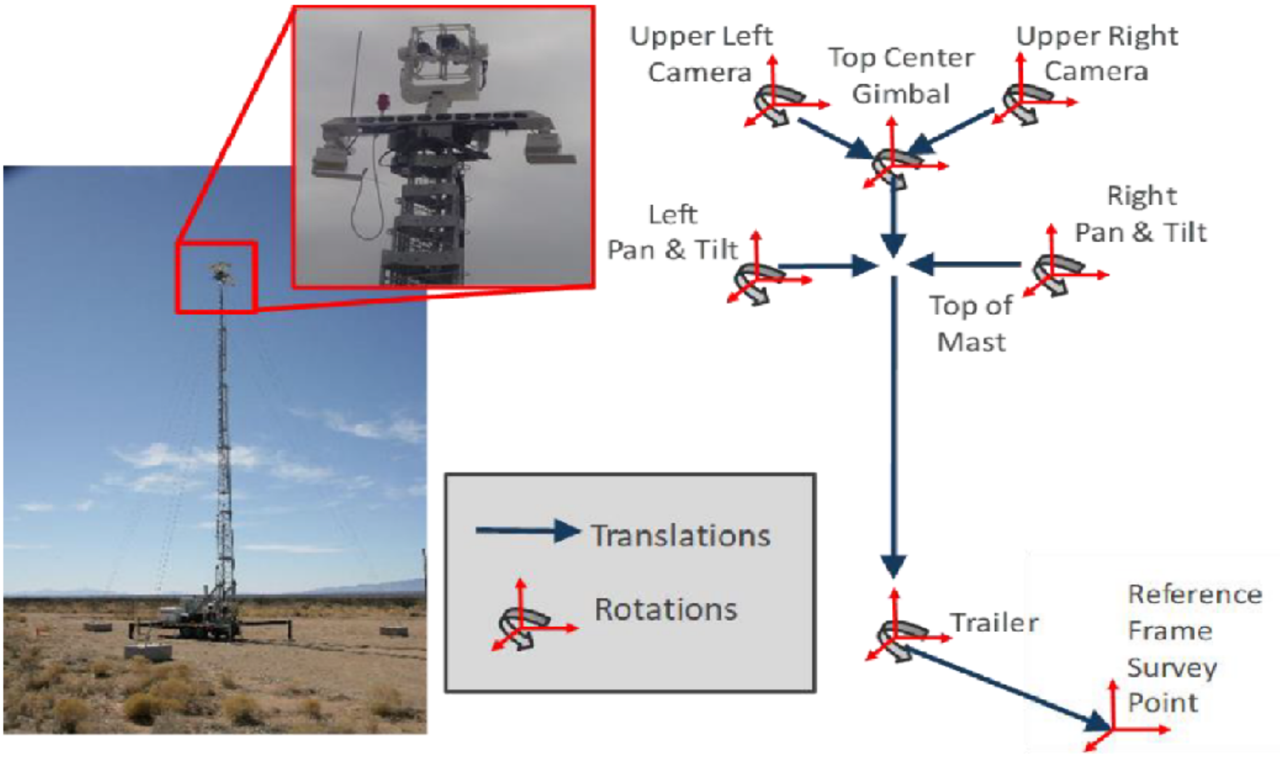
A UML model for the MISB Staging System is provided in Figure --.



This model only addresses a single stage. Non-trivial systems will require multiple stages. These stanges are assembled using the `parentStage` association of the `Stage` class. The following three concepts guide the construction of multiple-stage models:

- **Stage:** a single frame of reference located at a point. It defines the location, orientation, and kinemantics of a coordinate system located at that point. These properties can be defined in terms of absolute values, or as relative values measured from an "parent" reference system.
- **Constellation:** A system of one or more stages where the parent-child relationships between the stages is sufficient to calculate the absolute values for every stage.
- **Root Stage:** This is the starting point for a Constellation. This stage is expressed in absolute values (AbsGeocentric or AbsGeodetic).

An example articulated motion imagery system is illustrated in Figure --. The stages which describe this system are described in Table ---.



Example Motion Imagery System with Multiple Sensors and Gimbals (Photo credit White Sands Missile Range)

Table 1. Stage System Example

Stage	Component	Parent	Values
1	Reference Frame Survey Point	0	Absolute Position/Orientation
2	Trailer	1	Relative Position / Orientation
3	Left Camera	2	Relative Position / Orientation
4	Right Camera	2	Relative Position / Orientation
5	Top Center Gimbal	2	Relative Position / Orientation
6	Left Upper Camera	5	Relative Position / Orientation
7	Right Upper Camera	5	Relative Position / Orientation

The Staging System's absolute positions use either WGS84 Ellipsoid angular coordinates (i.e., geodetic Latitude, Longitude, Height above Ellipsoid (HAE)), or WGS84 geocentric Earth-Centered Earth-Fixed (ECEF) Cartesian coordinates (i.e., X, Y, Z).

The Staging System's relative positions use Cartesian coordinates (i.e., x, y, z) measured in meters from the parent stage frame of reference.

The Staging System's orientations use Euler rotations, measured in radians, around three axes (X, Y, and Z) of a righthanded coordinate system. The Staging System's absolute orientation has the X, Y, and Z axes aligned with the East, North, and Up (ENU) respectively.

The Staging System standardizes rotations to use a specific order of rotations. The Staging System uses Tait-Bryan angles with a Primary Rotation Order of Z-Y-X (see Appendix C); thereby the first rotation is around the Z-axis, the second rotation is around the new Y-axis (after rotation around the Z-axis), the final rotation is around the new X-axis (after rotation around the Z and then Y axes).

The Staging system does not explicity address temporality. However, this information is passed in the MISB KLV metadata stream. As such, each update is associated with the precision time stamp of that packet. In addition, an update can be associated with a Timer as described in section ----

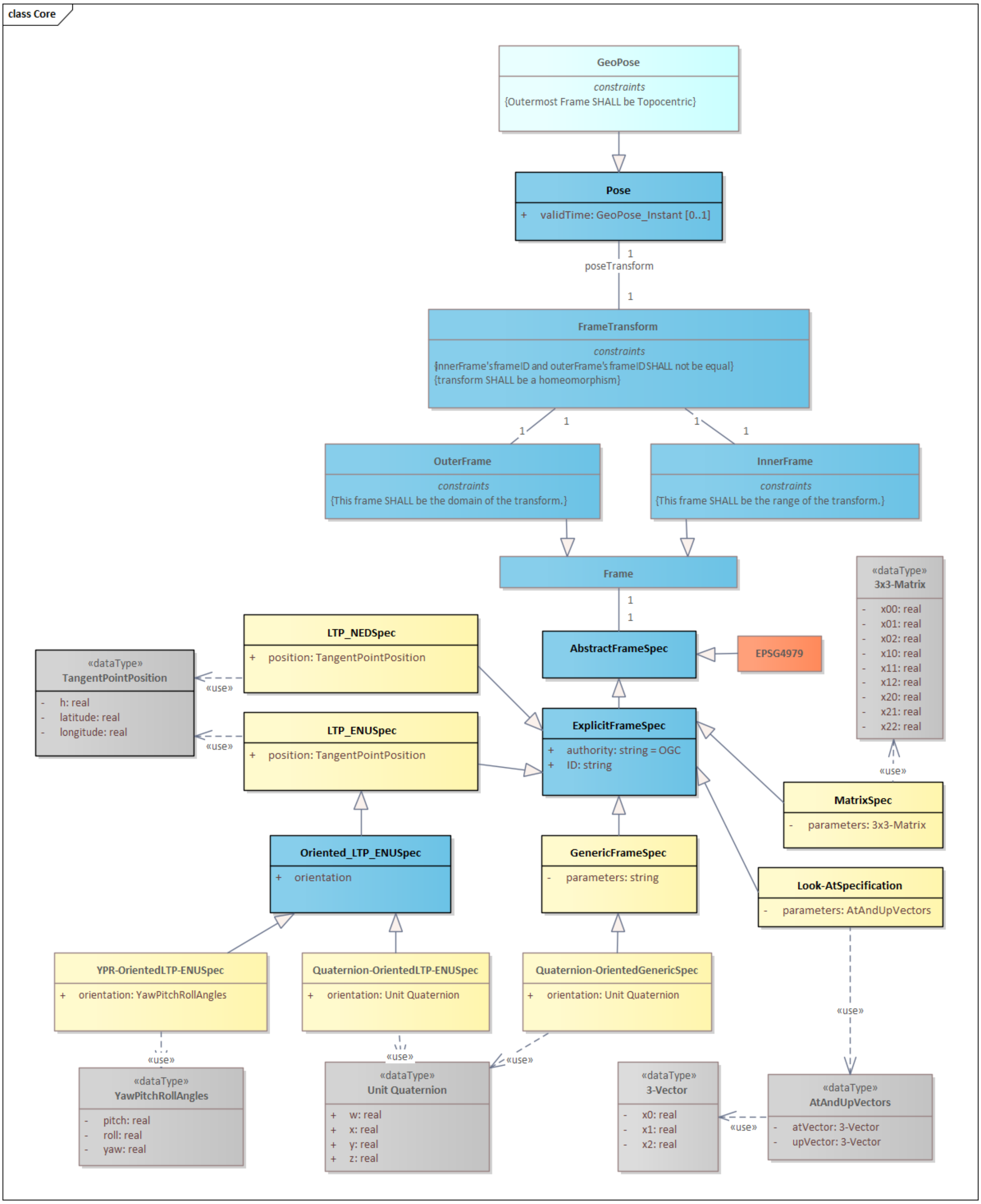
5.3.2. GeoPOSE

GeoPOSE is a proposed OGC standard which provides a general purpose alternative to the MISB Staging System.

This standard deals with the location and orientation of real or virtual geometric objects (Poses) an the aggregation of Poses into more complex structures.

Core

A UML model for the core GeoPOSE concepts is provided in figure ---



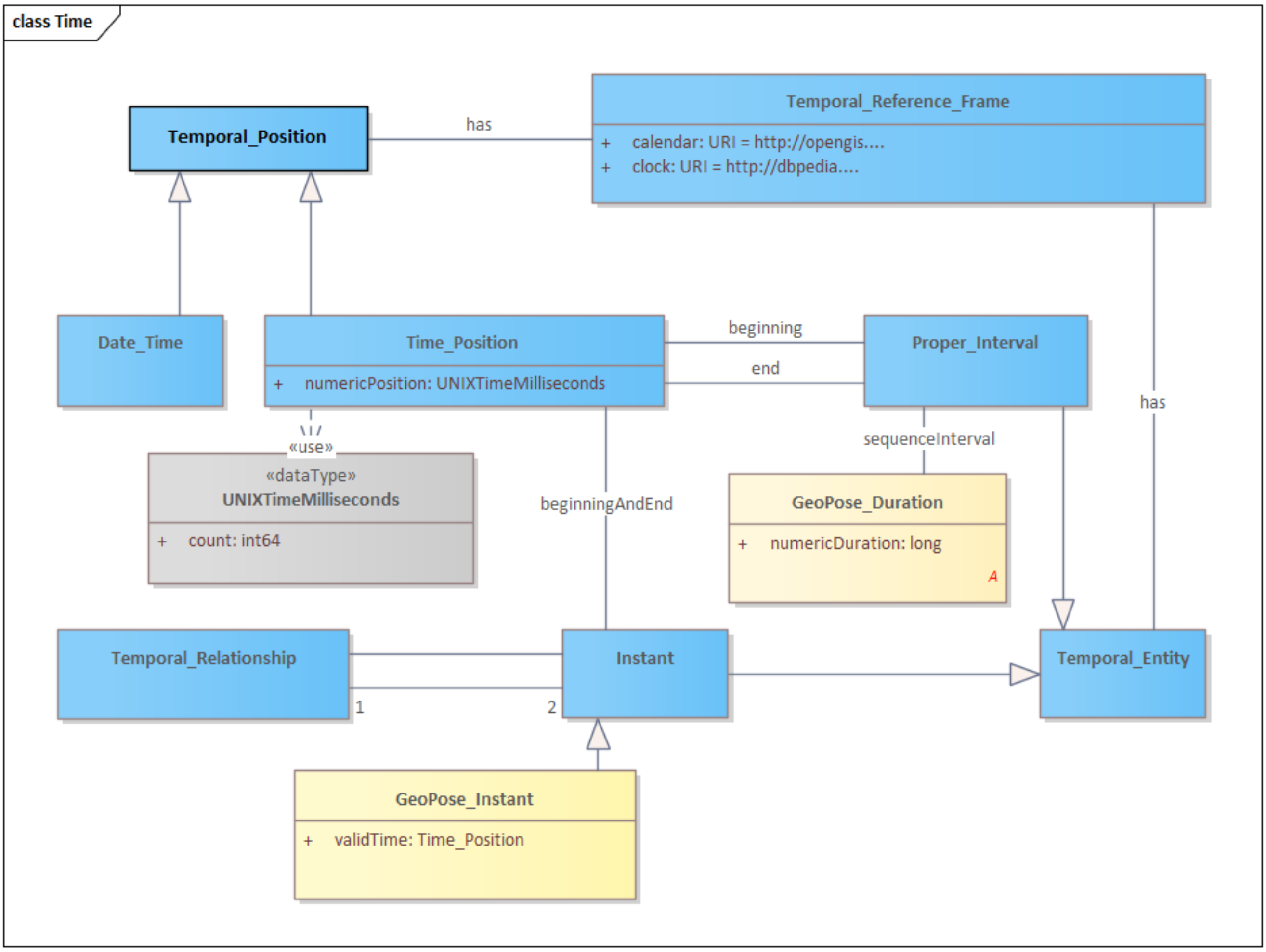
The key element in this model is the `FrameTransformation` class. This class expresses a transform between a pair of Reference Frames, Outer and Inner, each defined by a Frame Specification.

The Frame Transform is a representation of the transformation taking an Outer Frame coordinate system to an Inner Frame coordinate system. GeoPose v 1.0 supports transformations involving translation and rotation. The intention is to match the usual concept of a pose as a position and orientation.

Outer and Inner Frames are subclasses of the Frame Class. Frame provides a standard means of describing transforms for several common coordinate reference systems as well as a generic model.

Time

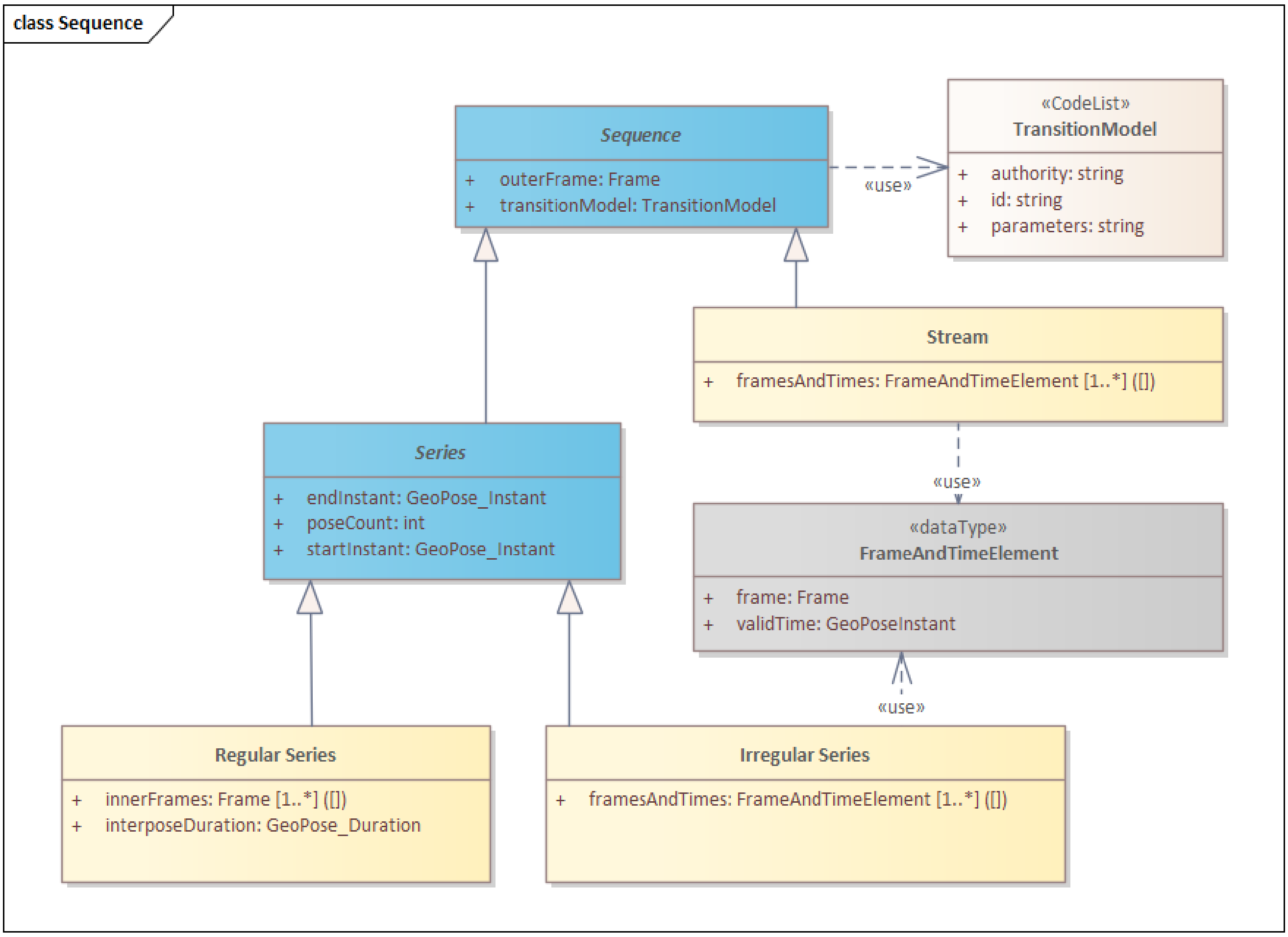
The GeoPOSE time model is rather simple. It does away with calendard and restricts time positions to milliseconds of UNIX Time.



The most important classes are GeoPose_Instant (a single point in time) and GeoPose_Duration (a period of time).

Sequence

The sequence logical model defines a method for the packaging of GeoPose data, where multiple GeoPoses in a sequence share the same Outer Frame and there is a time-dependent changing Inner Frame.



The GeoPOSE Standard provides three models for packaging a collection of Inner Frames.

- Stream: the Inner Frame definition (Frame) and an associated time stamp are delivered sequentially.
- Reqular Series: the Inner Frame definitions are delivered as a sequence, separated by a fixed time interval.
- Irregular Series: the Inner Frame definitions and associated time stamps are delivered as a collection. There is no explicit spatial or temporal order to the frames.

5.3.3. Discussions

Quaternions vs. Euler rotations

Temporal representation - Unix time vs the Timer model.

Standarize on a right-handed Cartisian coorinate system for all stages/Poses?

Common concepts:

1. An "anchor" node which ties the relative coordinates to an absolute CRS
2. Local coordinate systems associated with each other through standardized transformations

5.4. Mass Properties

Consider an articulated vehicle in free-flight. As the components change position, they change the center of mass, priciple axis of rotation, and a number of other mass properties. These changes affect the dynamic behavior of the vehicle. Most designers take great care to assure that problems cannot occur. But that doesn't mean we should ignore the issue.

Comments?

5.5. Temporal Reference Systems

Proposition: Time is in the eye of the beholder. So all measurements of time must be local.

Dynamic Features are not tied to an Earth-centered static existance. Yet the concepts of time used in the geospatial community are almost exclusivly based on Earth-centric astronomical phenomena. They also assume a rather coarse (days, weeks, years) degree of granularity. For dynamic features we need to use local clocks with precision down to the nanosecond. We are less concerned with absolute time than with relative time. State B was achieved 37 nanoseconds after State A.

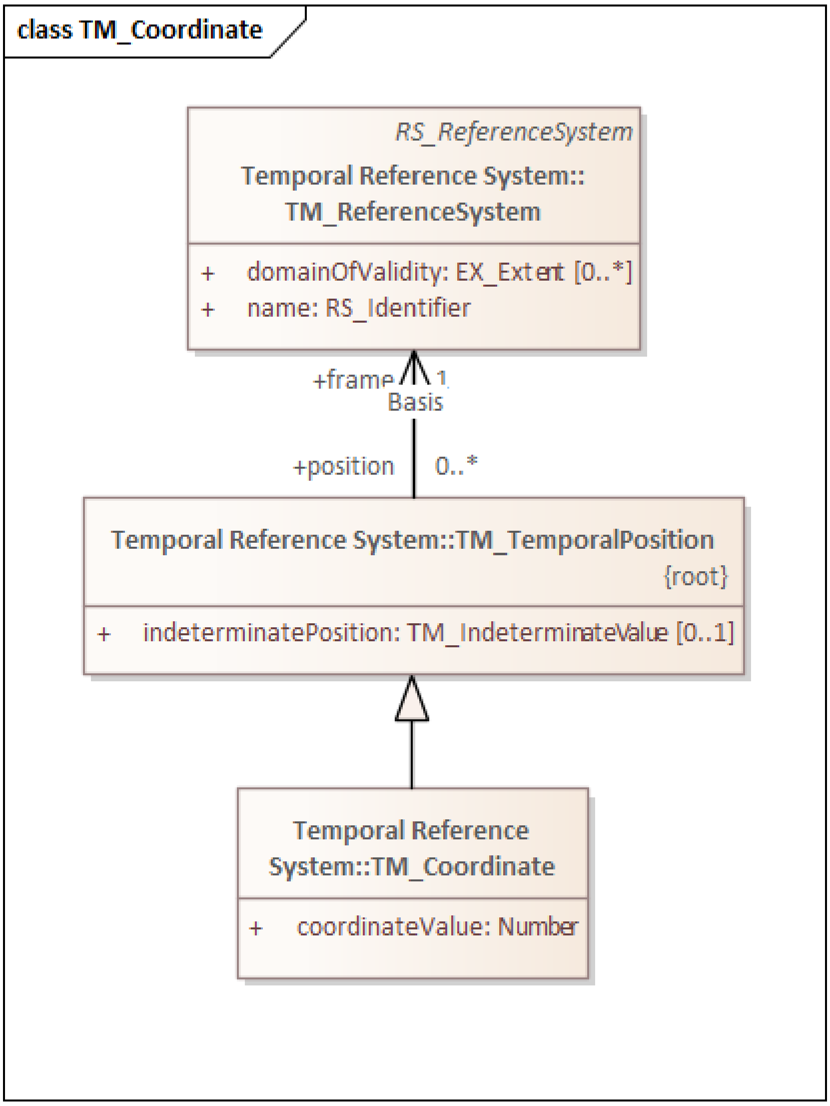
It's only when we begin aggregating these dynamic elements that we begin to worry about "absolute" time. Even then, we are more likely to convert from one local clock to another then to convert to an absolute time.

So if all time is local, we need a Temporal Reference System concept which captures the parameters needed to transform across TRSs. A temporal equivalent to GeoPOSE.

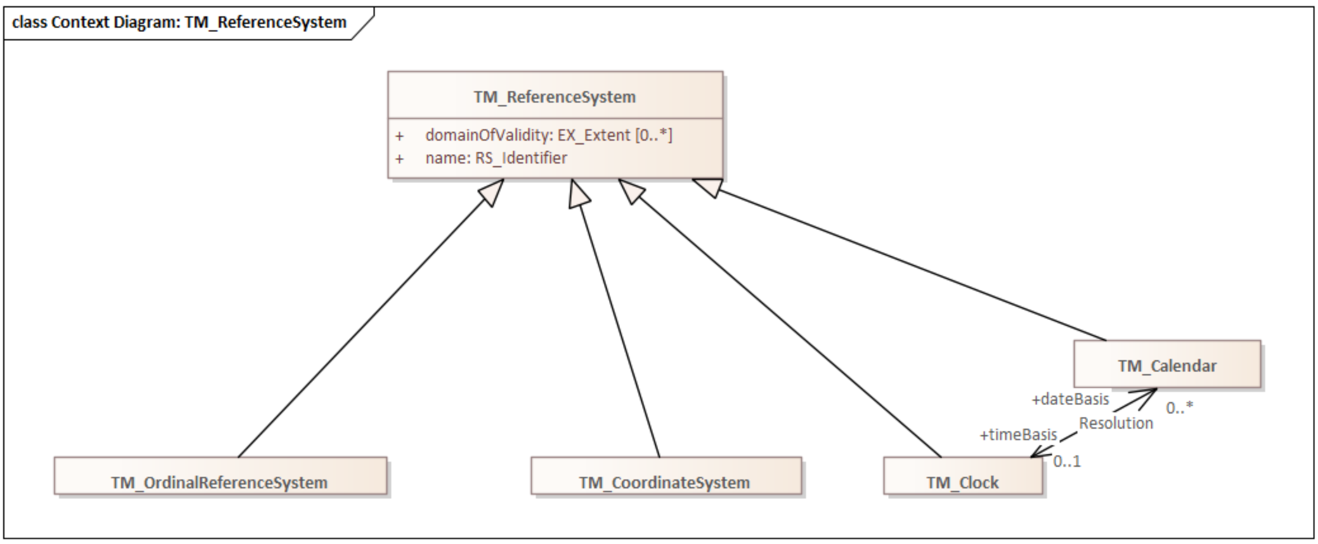
5.5.1. ISO 19108

The ISO standard for Temporal Reference Systems is ISO 19108:2006 Geographic Schema - Temporal Schema.

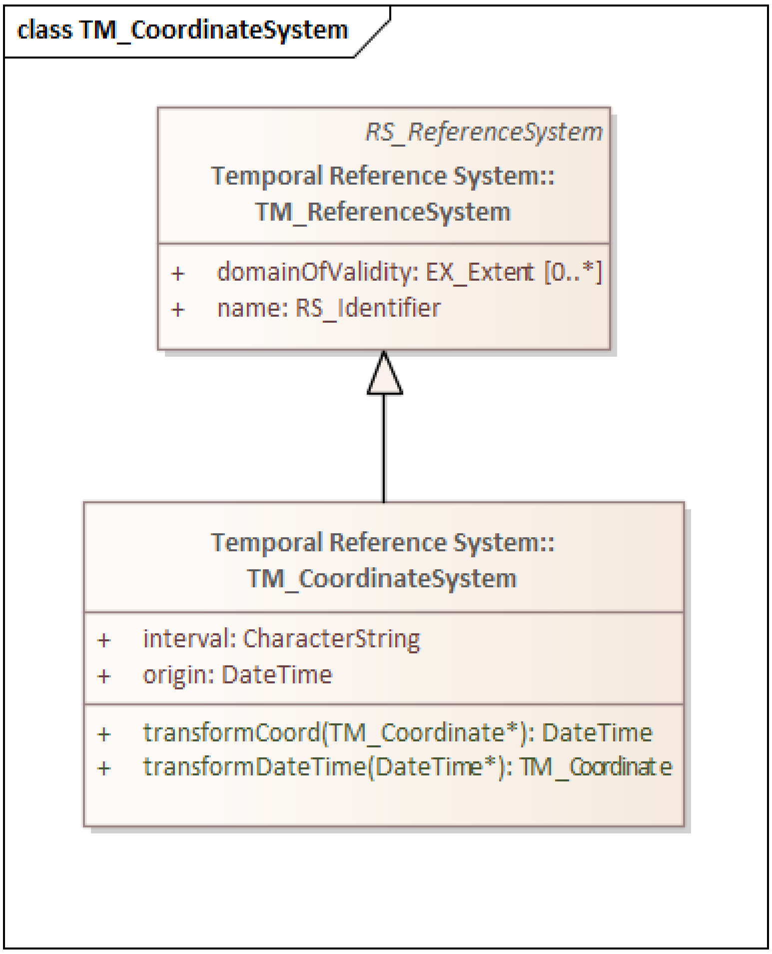
In our discussion of Moving Features, we found that time-variant properties of Features were selected using a TM_Coordinate parameter. TM_Coordinate is defined in ISO 19108 and illustrated in Figure ---



From this fiqure we see that a TM_Coordinate is a type of TM_TemporalPosition. And that each TM_TemporalPosition is associated with the TM_ReferenceSystem within which it is measured. So to understand Temporal Reference Systems as defined in ISO 19108, we must explore the capabilites and limits of TM_ReferenceSystem.



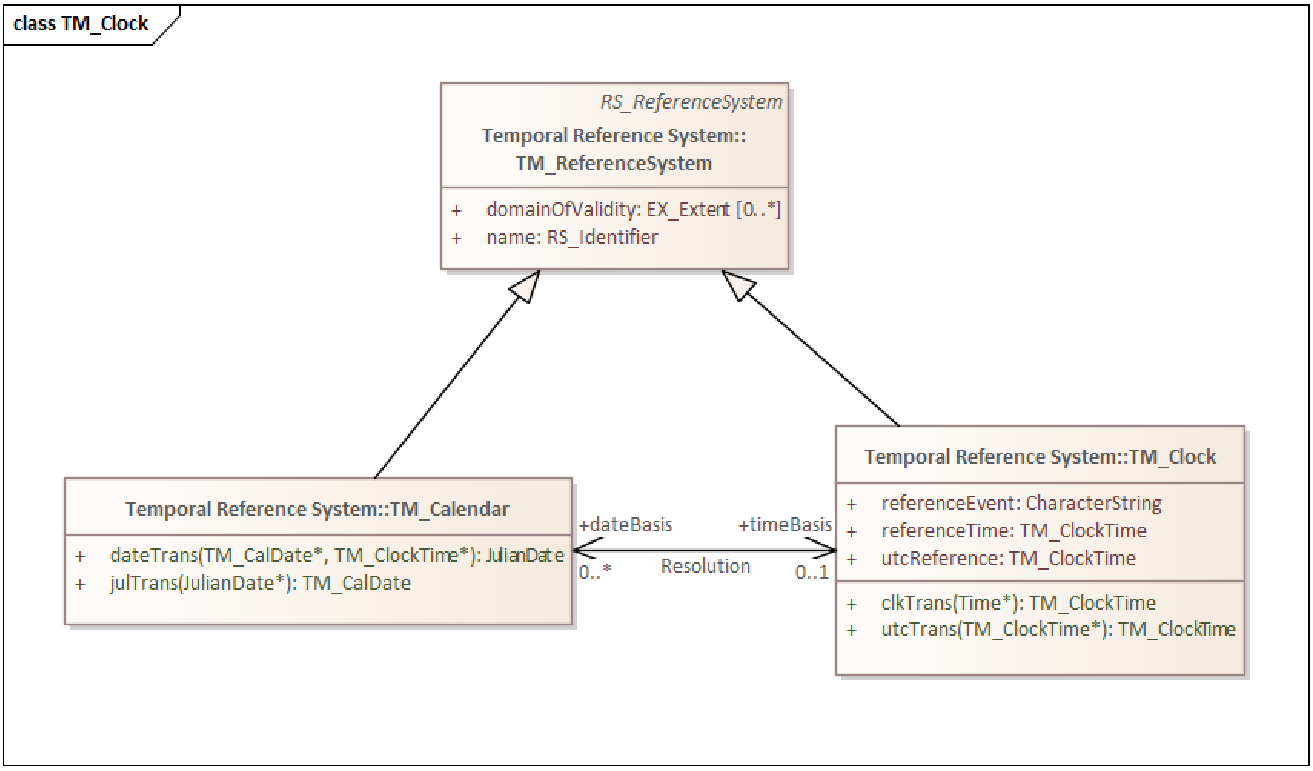
We see from Figure — that there are four types of TM_ReferenceSystems. Since our concern is with time, not date, the TM_CoordinateSystem and TM_Clock variants are of most interest.



TM_CoordinateSystem is "A system for measuring time on a continuous interval scale using a single standard time interval". The standard time interval is provided through the `interval` attribute. In addition, the `origin` attribute provides a temporal "datum" from which time is measured. Since time is a one-dimensional quantity, the origin and interval are sufficient to define a basic Temporal Coordinate Reference System.

However, TM_CoordinateSystem does have limitations. The transformation operations, for example, only convert to and from DateTime. DateTime is a primitive type defined in ISO 19103. These operations cannot be used to convert from one TM_CoordinateSystem to another. Furthermore. TM_CoordinateSystem does not provide sufficient information to perform these transformations trough an outside service.

The other alternative is TM_Clock illustrated in Figure —

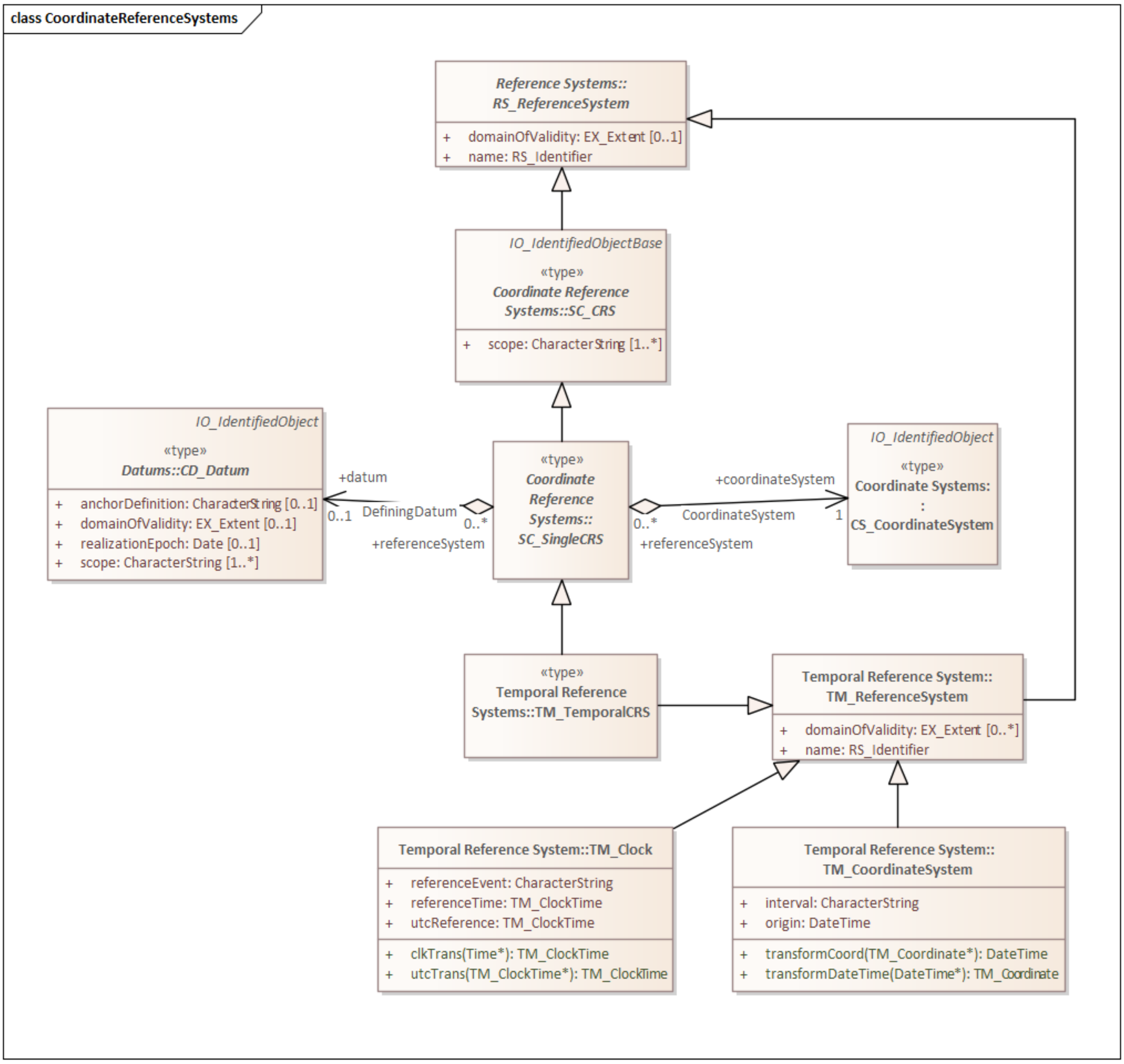


TM_Clock is "A system for measuring temporal position within a day". It has an optional `dateBasis` association with a calendar (TM_Calendar).

This combination of classes allows us support high precision local-clock TRS as well as full date-time TRS.

To be provided - the main limitation is its close association with TM_Calendar.

At this point it appears that transformations between clocks is not supported by 19108. But there is another avenue we can explore. TM_CoordinateSystem is a subclass of TM_ReferenceSystem. And TM_ReferenceSystem is a subclass of RS_ReferenceSystem. This takes us into ISO 19111: Referencing By Coordinates.



In this figure, we see that a TM_ReferenceSystem is an RS_ReferenceSystem as defined in ISO 19111. But it is not an SC_CRS. As such, it does not inherit the additional properties defined in the subclasses of SC_CRS. But we do have a way out. TM_TemporalCRS (defined in 19111 of all places) is a subclass of both SC_SingleCRS (19111) and TM_ReferenceSystem (19108). So if we want to promote Time to a first-class dimension, then TM_TemporalCRS is a good place to start.

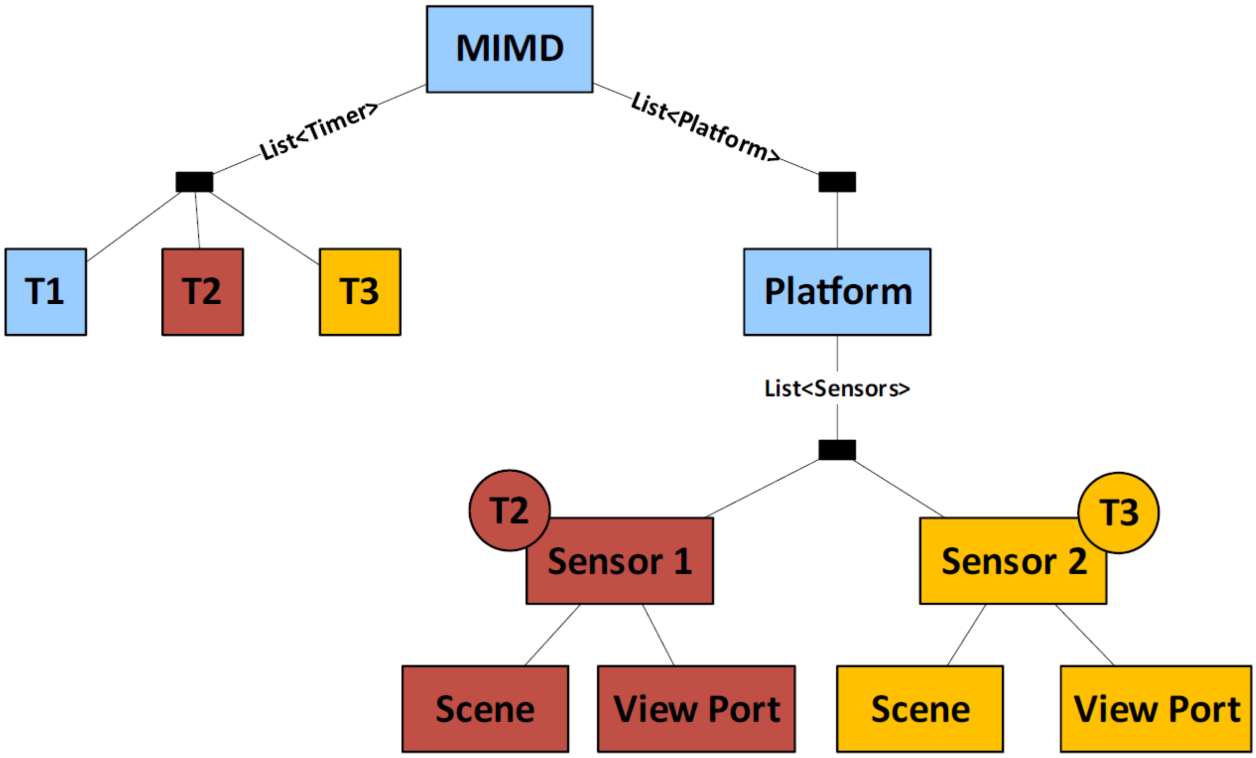
The question is, what do we need from a temporal coordinate reference system?

5.5.2. MISB Timers

Timing is critical for motion imagery. Not only are accurate time stamps essential for the smooth viewing of the images, they are even more critical for the accuracy of data derived from that imagery. Proper geolocation of a moving image, for example, requires nano-precision time stamps.

For that reason, the Motion Imagery Standards Board (MISB) developed a timing infrastructure for use with MISB standards. This infrastructure addresses not only the locality of time, but also the conversion of time values from one locality to another.

Three timers are illustrated in the example below (T1, T2, and T3). T1 is associated with the Platform. This timer receives regular GPS updates and is considered the most accurate in this system. T2 is associated with Sensor 1. This is a simple oscillator, counting nanoseconds since the sensor was last powered up. T3 is associated with Sensor 2. It also is a simple oscillator.



While timers T2 and T3 are sufficient for the operation of their associated sensors, data based on time values from these Timers would not be usable. A transformation from the local time to global time is required.

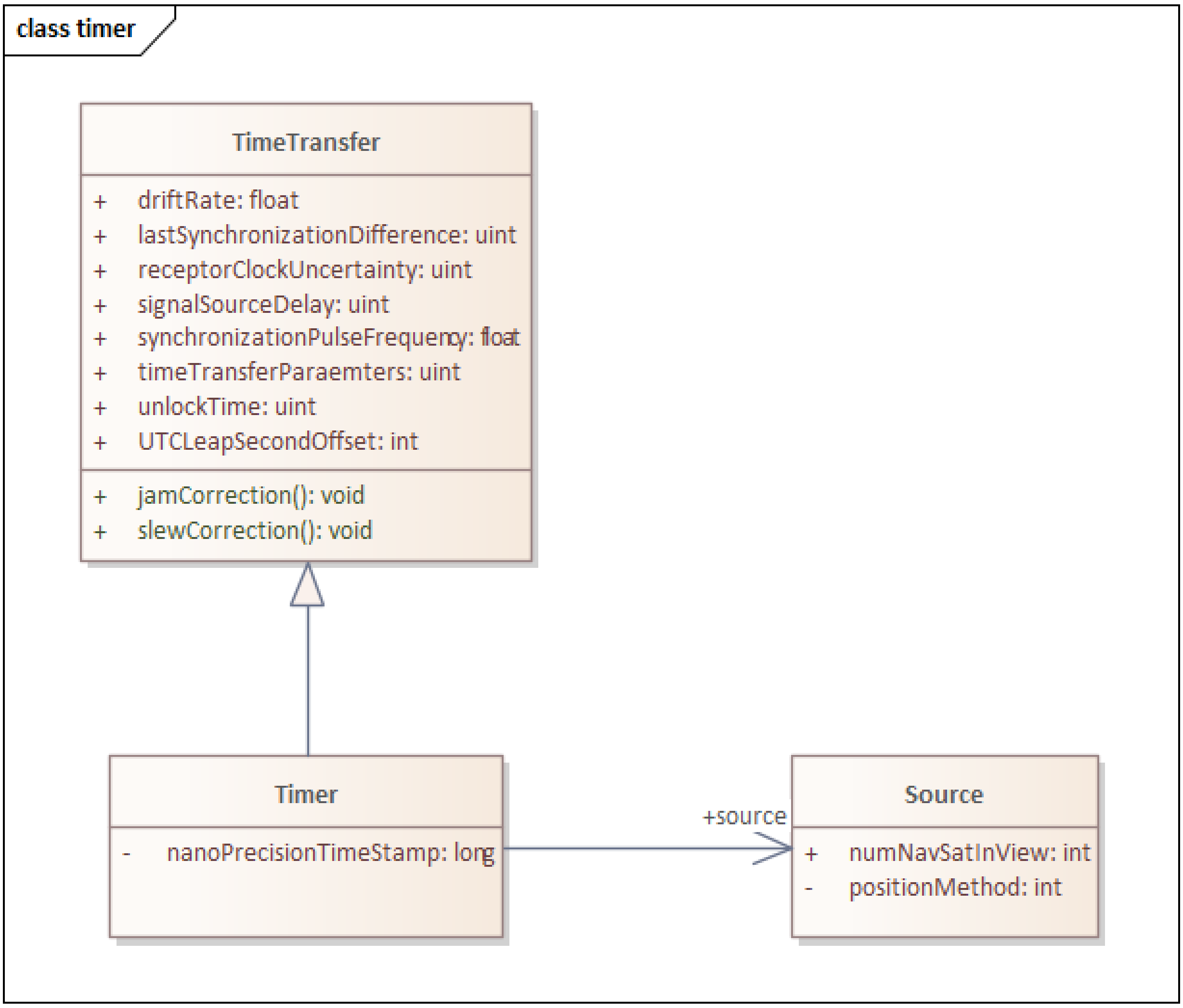


figure — illustrates the Timer architecture. A Timer is a physical device which provides a time stamp with up to nanosecond precision. The timer is associated with a "Source". The Source specifies how the timer is initallized and any subsequent corrections are applied. Finally, the TimeTransfer pack defines the metadata avaiable to detect and apply any corrections. to the time stamp.

The MISB timer concept is specific to the platforms and collection techniques used for Motion Imagery. These can be extended to provide a more general solution.

5.6. Dynamic Features at Relativistic Velocities

At very high velocities, measurements between moving features sufer from relativistic effects. The proposition is to build on Minkowski SpaceTime to address these cases.

5.6.1. Lorentz Transformation

A transformation between two different coordinate frames that move at a constant velocity and are relative to each other.

ONly related to changes in inertial reference frames:

- Inertial frames - motion with a constant velocity

- Non-inertial frames - rotational motion with constant angular velocity, acceleration in curved paths.

In the reference frame "F" which is stationary, the coordinate defined are x, y, z, and t. In another reference frame F' which moves at a velocity v which is relative to F and the observer defines coordinate in the moving reverence frame as x', y', z', t'. In both the reference frames the coordinate axis are parallel and they remain mutually perpedicular. The relative motion is along the xx' axes. At t = t' = 0, the origins in both reference frames are the same (x,y,z) = (x',y',z') = (0,0,0).

The Lorentz factor g = 1 / (sqrt(1 - (v^2 / c^2)))

x' = g(x-vt) t' = g(t-(vx/c^2)) y' = y z' = z

This is not a complete coordinate transformation since F' has to be rotated and translated so as to be co-linear with F. However, it does add the impact that relative velocity (v) has on the measurements of x and t. In most cases this impact is neglegible (v^2 / c^2 approaches 0). However, when v is a significant percentage of c it should be applied.

5.6.2. Minkowski Space Time

It is basically a combination of 3-dimensional Euclidean Space and time into a 4-dimensional manifold, where the interval of spacetime that exists between any two events is not dependent on the inertial frame of reference.

Minkowski spacetime is a 4-dimensional coordinate system in which the axes are given by (x, y, z, ct)

Where ct is time (t) times the speed of light (c)

ds^2 = $-c^{2dt}2$ + dx^2 + dy^2, + dz^2 = the differential arc length in space time where:

1. dt = change in time
2. dx = change in x direction
3. dy = change in y direction
4. dz = change in z direction

Key point - while a Lorentz transformation deals with spatial measurements, Minkowski space includes time as part of that space-time. Thus ds is an arc length through space-time as opposed to a difference in x as in the Lorentz transform.

Question, since $c^{2dt}2$ is a negative term, does that inply that ct is an imaginary number orthagonal to x, y, and z (cti) such that i^2 = -1?

Yes for complex Minkowski space time. Here it is expressed as x^2 + y^2 + z^2 + (ict)^2 = const.

Complex Minkowski spacetime was replaced with real Minkowski space time where time is a real coordinate rather than an imaginary one.

Where v is velocity, and x, y, and z are Cartesian coordinates in 3-dimensional space, and c is the constant representing the universal speed limit, and t is time, the four-dimensional vector v = (ct, x, y, z) = (ct, **r**) is classified according to the sign of $(c^2 t^2) - r^2$. A vector is **timelike** if $(c^2 t^2) > r^2$, **spacelike** if $(c^2 t^2) < r^2$, and **null** or **lightlike** if $(c^2 t^2) = r^2$. This can be expressed in terms of the sign of η(v, v) as well, which depends on the signature. The classification of any vector will be the same in all frames of reference that are related by a Lorentz transformation (but not by a general Poincaré transformation because the origin may then be displaced) because of the invariance of the interval.

Annex A: Bibliography

Hughes, P. :Spacecraft Attitude Dynamics, Dover Publications Inc. 2024

ISO/TC 211: ISO 19107:2019 Geographic information - Spatial schema

ISO/TC 211: ISO 19107:2003 Geographic information - Spatial schema

ISO/TC 211: ISO 19109:2005 Geographic information - Rules for application schema

ISO/TC 211: ISO 19109:2005 Geographic information - Rules for application schema

ISO/TC 211: ISO 19141:2008 Geographic information - Schema for moving features

OGC: OGC 20-010, OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard (2021)

OGC: OGC 19-011r4, OGC IndoorGML 1.1 (2020)

OGC: OGC 19-045r3, OGC MOving Features Encoding Extension - JSON (2020)

Annex B: Revision History

Date	Release	Editor	Primary clauses modified	Description
2016-04-28	0.1	G. Editor	all	initial version