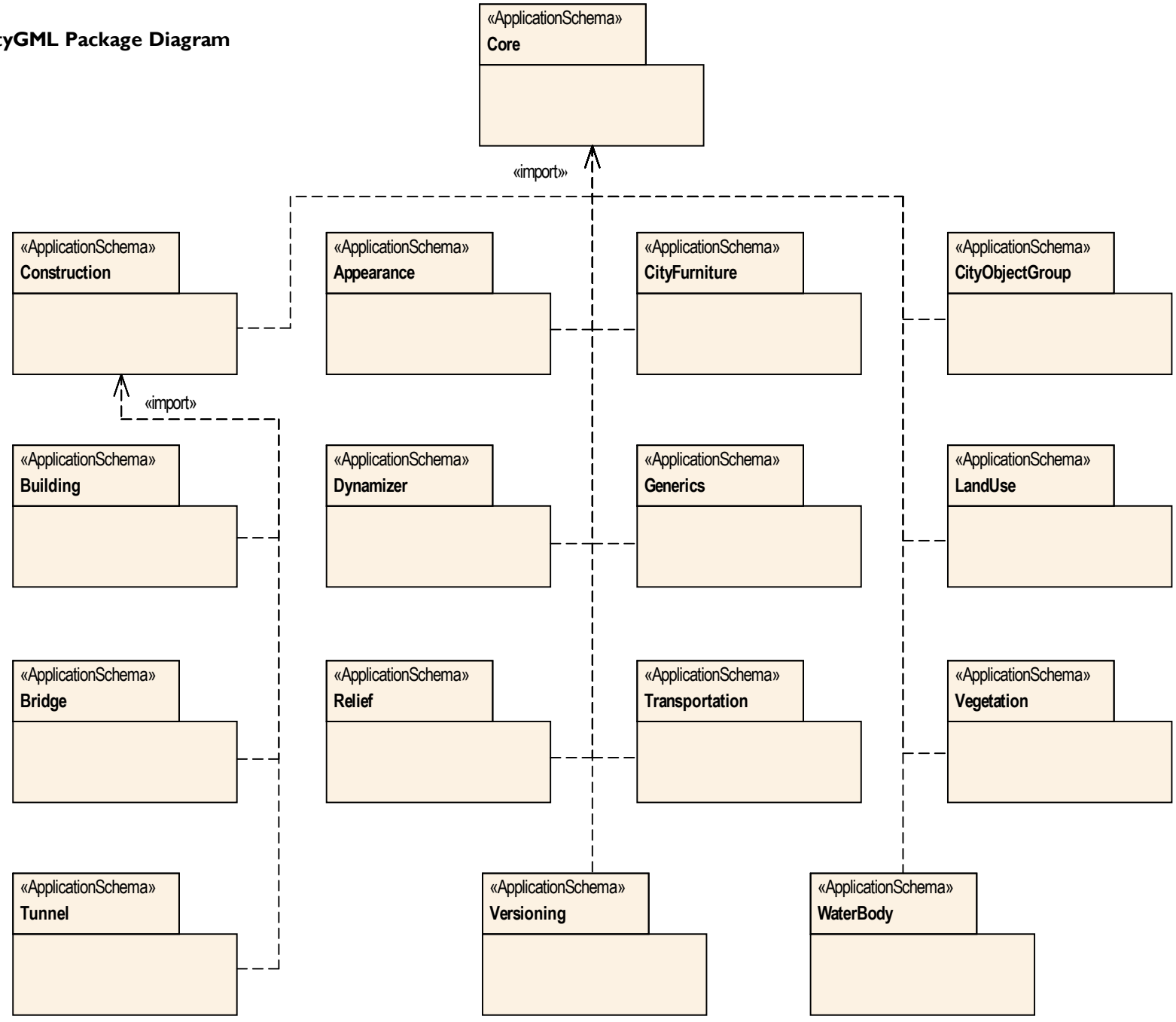
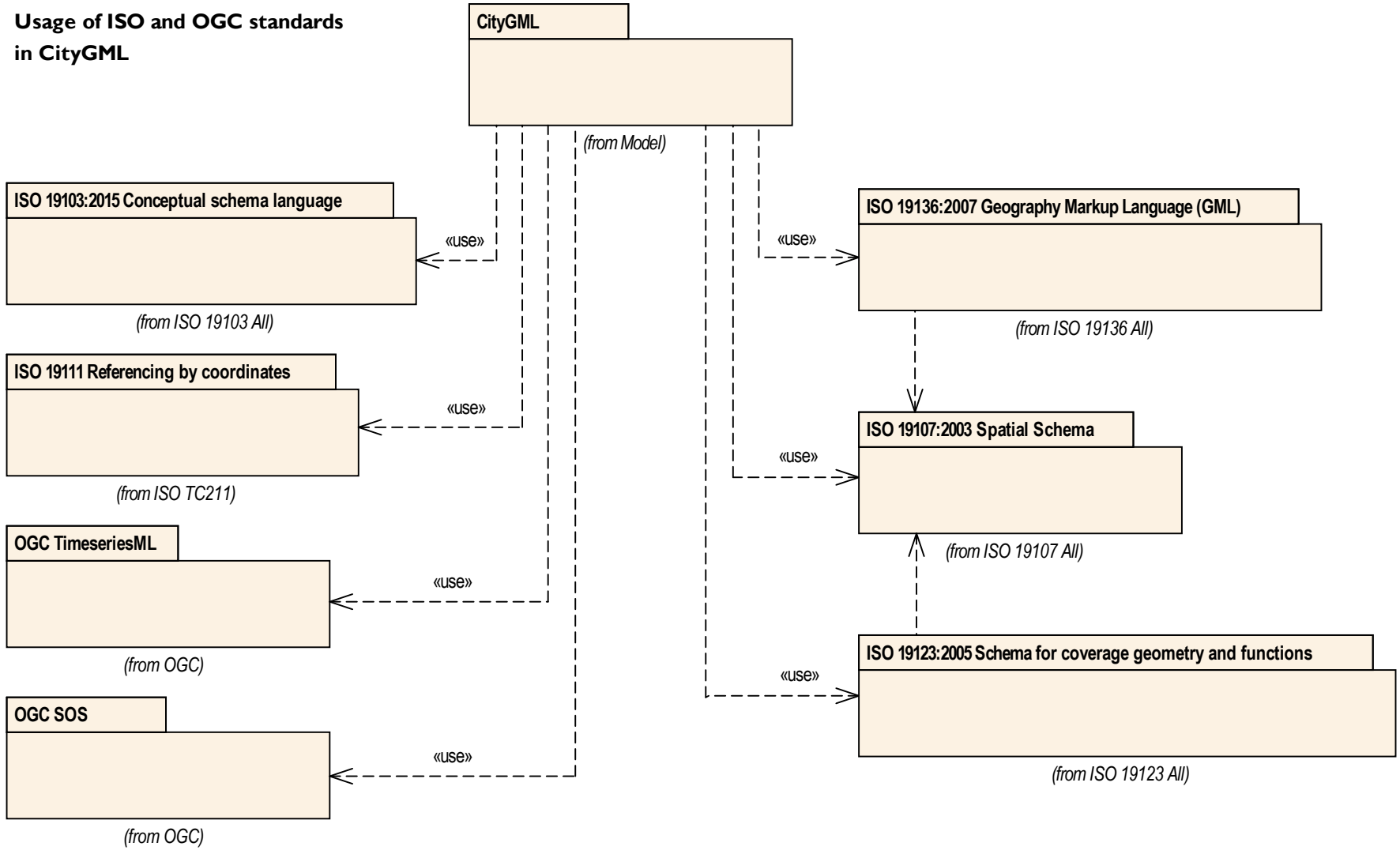
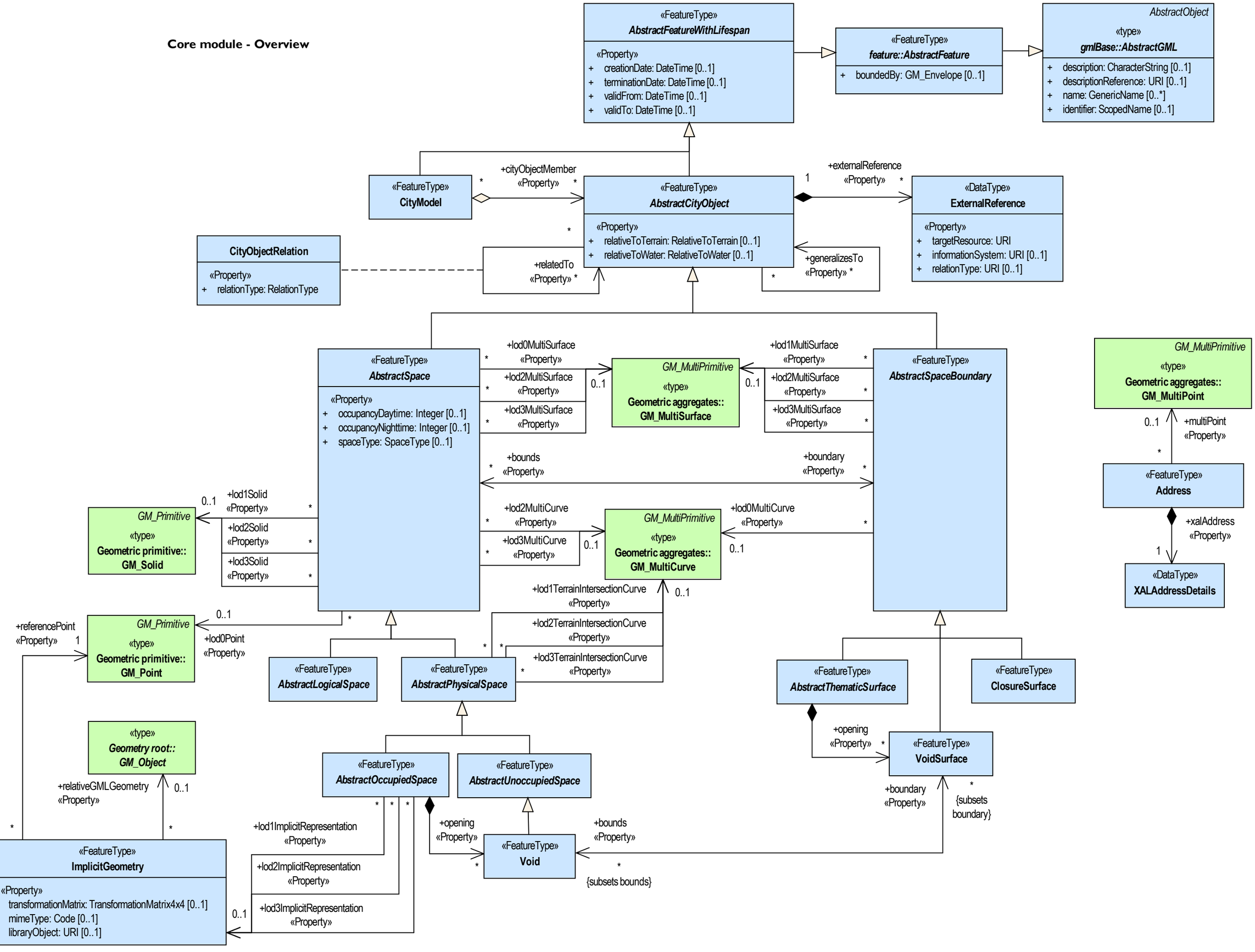
CityGML Package Diagram



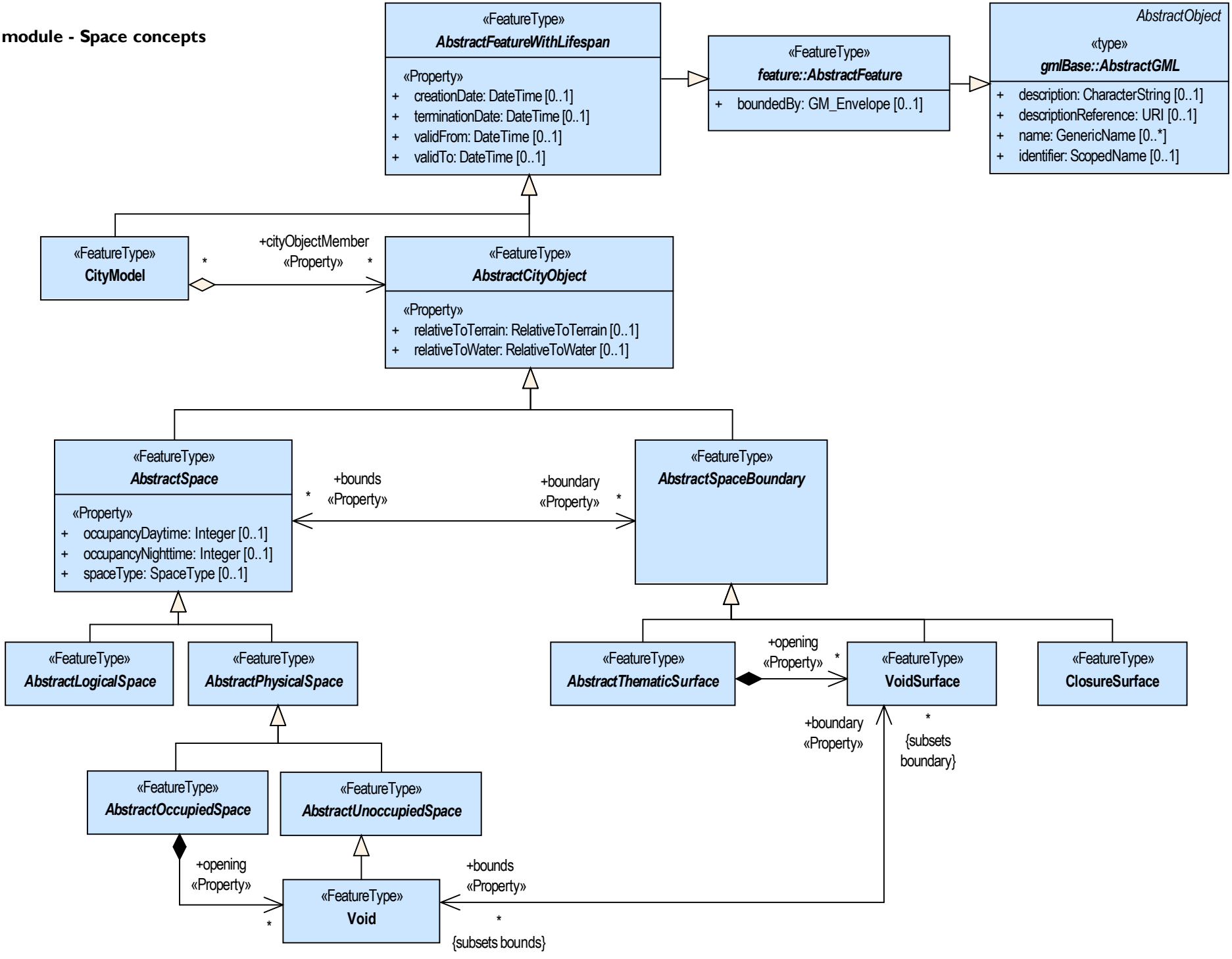
Usage of ISO and OGC standards in CityGML



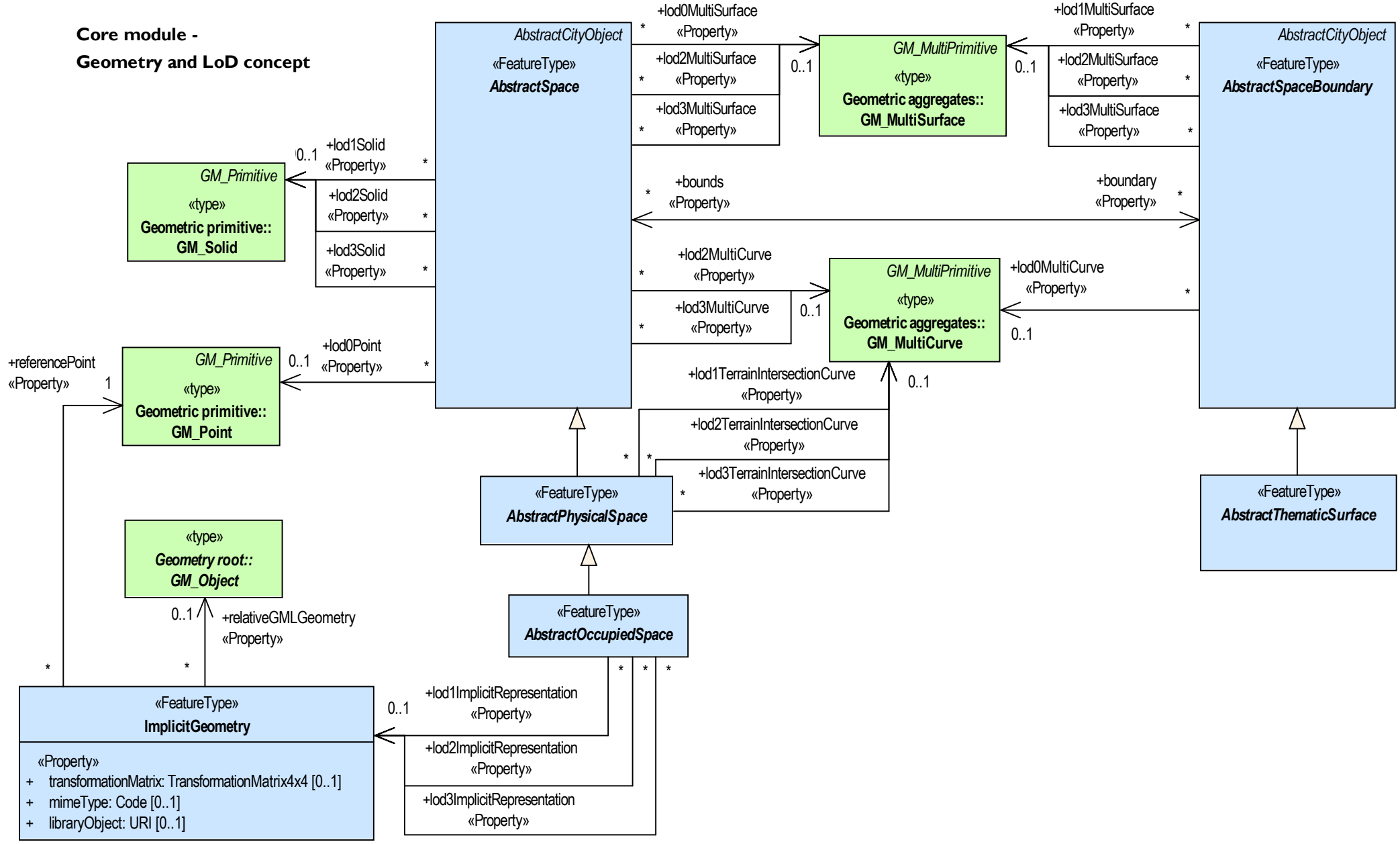
Core module - Overview



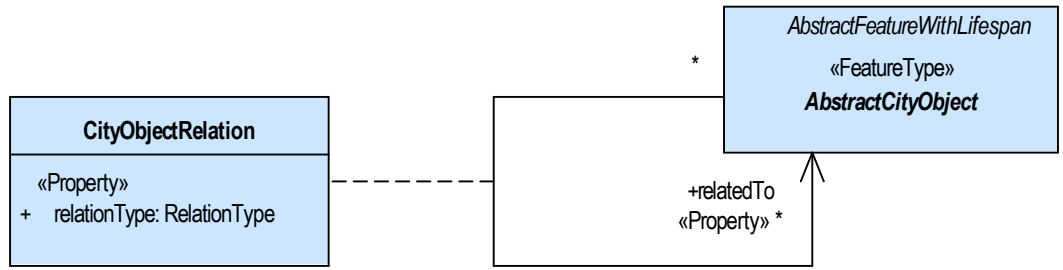
Core module - Space concepts

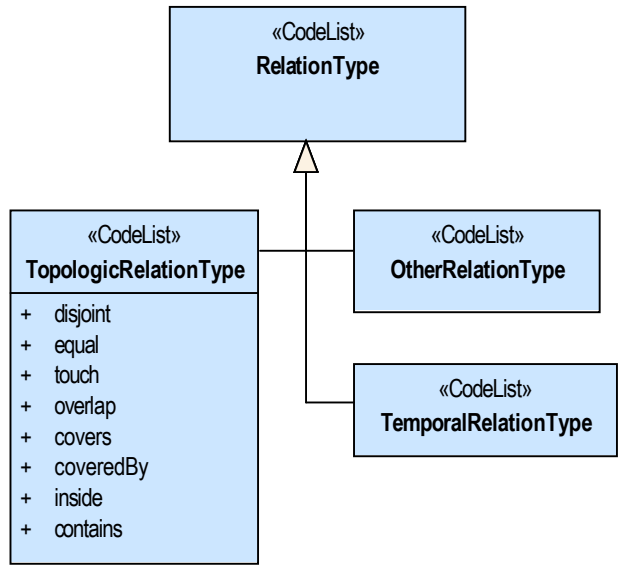


Core module -
Geometry and LoD concept

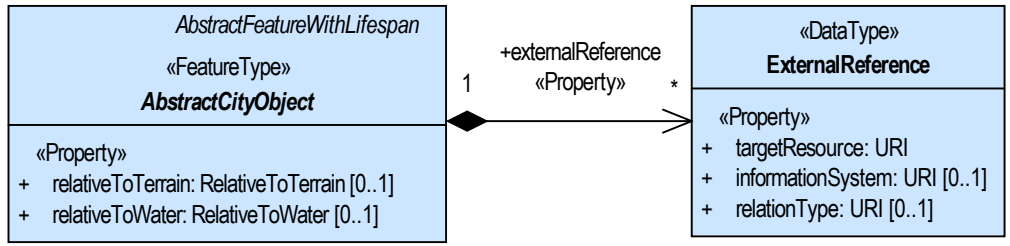


Core module - City object relations



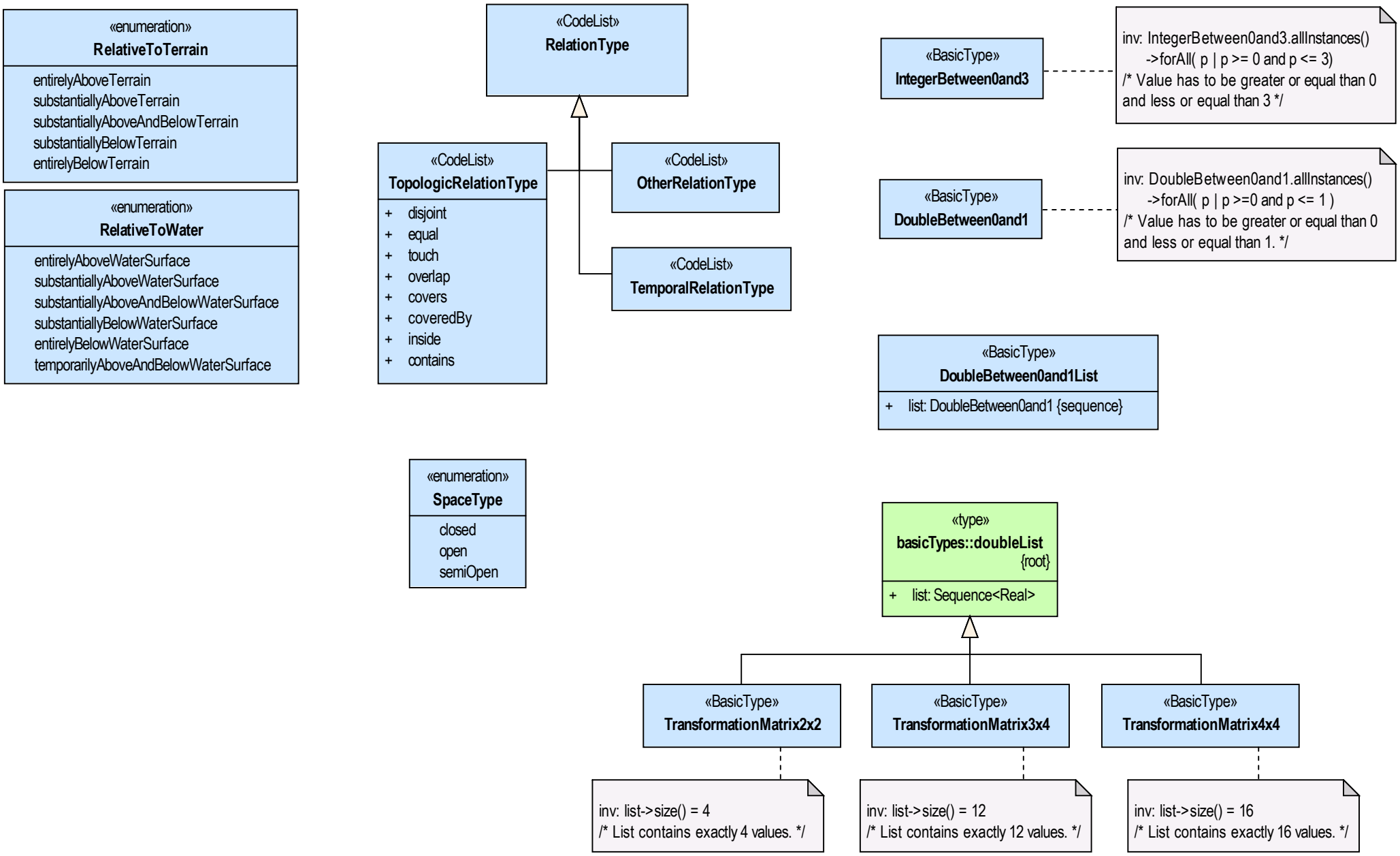


Core module - Miscellaneous

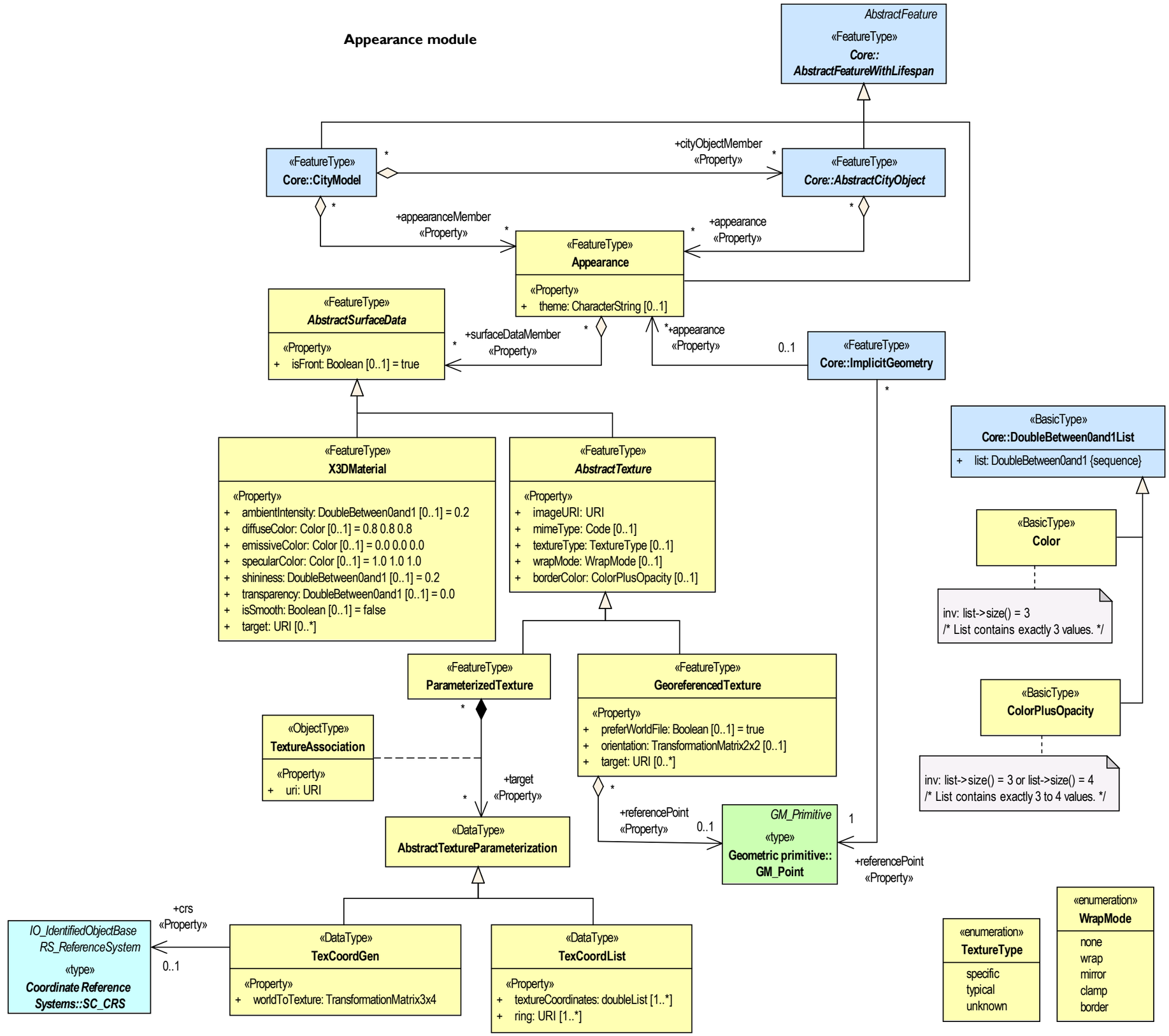


ExternalReference is now extended by an optional ***relationType*** which can link to some external definition of the type of relation (e.g. the *sameAs* relation from OWL). Hence, ***ExternalReferences*** can now be used to express relations similar to RDF.

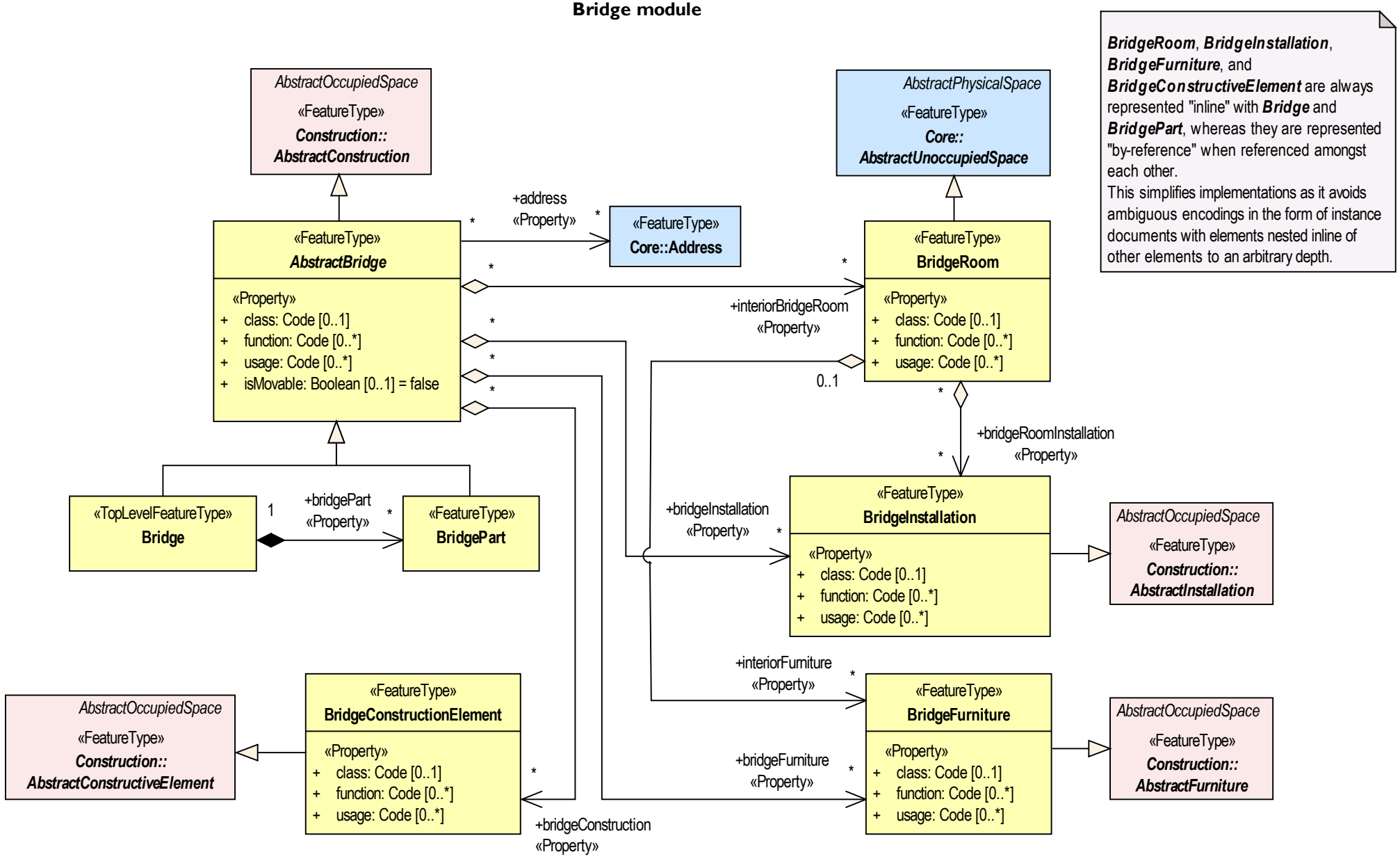
Core module - Basic Types and Enumerations



Appearance module

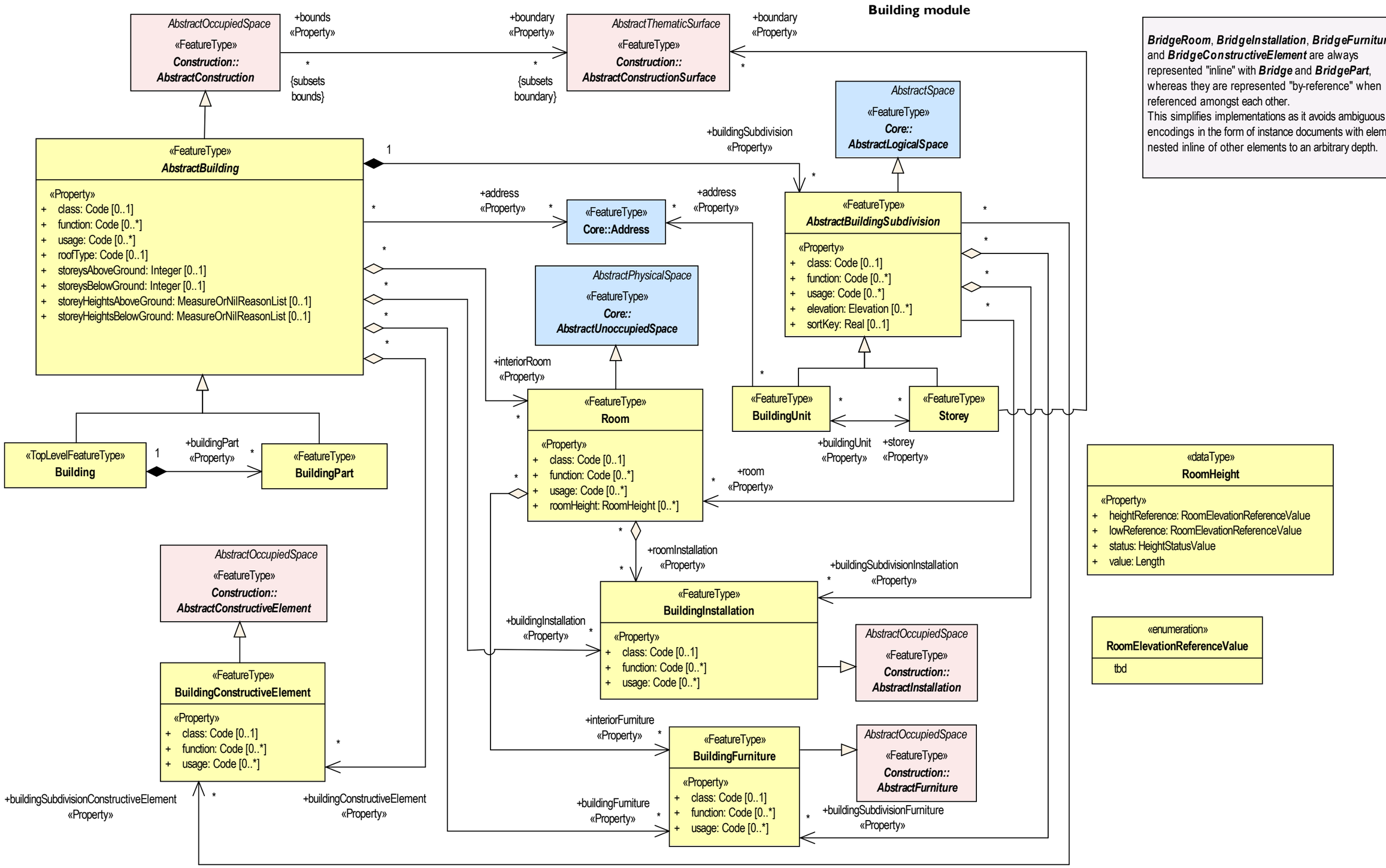


Bridge module



BridgeRoom, ***BridgeInstallation***, ***BridgeFurniture***, and ***BridgeConstructiveElement*** are always represented "inline" with ***Bridge*** and ***BridgePart***, whereas they are represented "by-reference" when referenced amongst each other.
This simplifies implementations as it avoids ambiguous encodings in the form of instance documents with elements nested inline of other elements to an arbitrary depth.

Building module



BridgeRoom, ***BridgeInstallation***, ***BridgeFurniture***, and ***BridgeConstructiveElement*** are always represented "inline" with ***Bridge*** and ***BridgePart***, whereas they are represented "by-reference" when referenced amongst each other.
This simplifies implementations as it avoids ambiguous encodings in the form of instance documents with elements nested inline of other elements to an arbitrary depth.

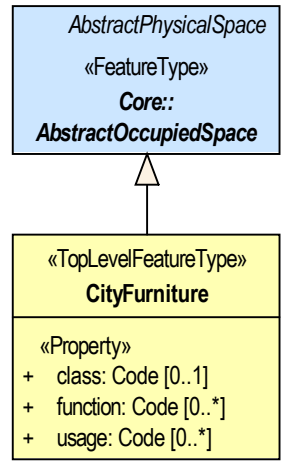
«dataType»
RoomHeight

«Property»
+ heightReference: RoomElevationReferenceValue
+ lowReference: RoomElevationReferenceValue
+ status: HeightStatusValue
+ value: Length

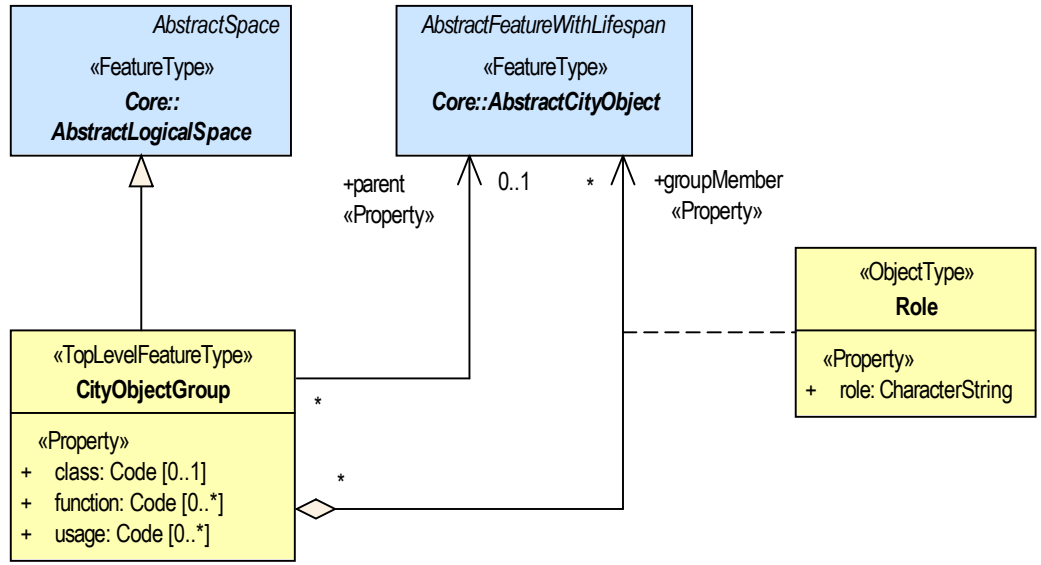
«enumeration»
RoomElevationReferenceValue

tbd

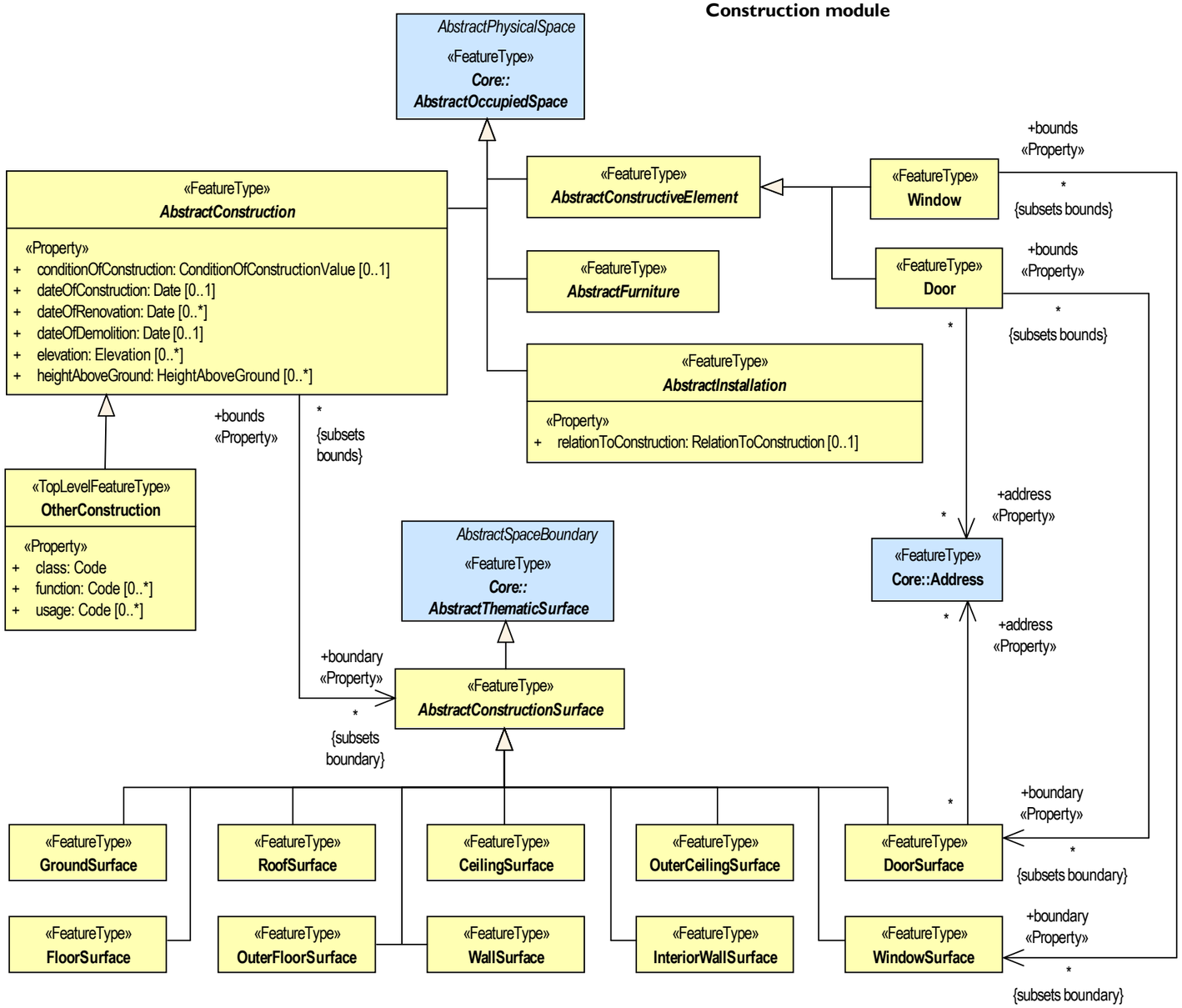
CityFurniture module



CityObjectGroup module



Construction module



«enumeration»
ConditionOfConstructionValue

- declined
- demolished
- functional
- projected
- ruin
- underConstruction

«enumeration»
RelationToConstruction

- inside
- outside
- bothInsideAndOutside

«DataType»
Elevation

«Property»
+ elevationReference: ElevationReferenceValue
+ elevationValue: DirectPosition

«DataType»
HeightAboveGround

«Property»
+ heightReference: ElevationReferenceValue
+ lowReference: ElevationReferenceValue
+ status: HeightStatusValue
+ value: Length

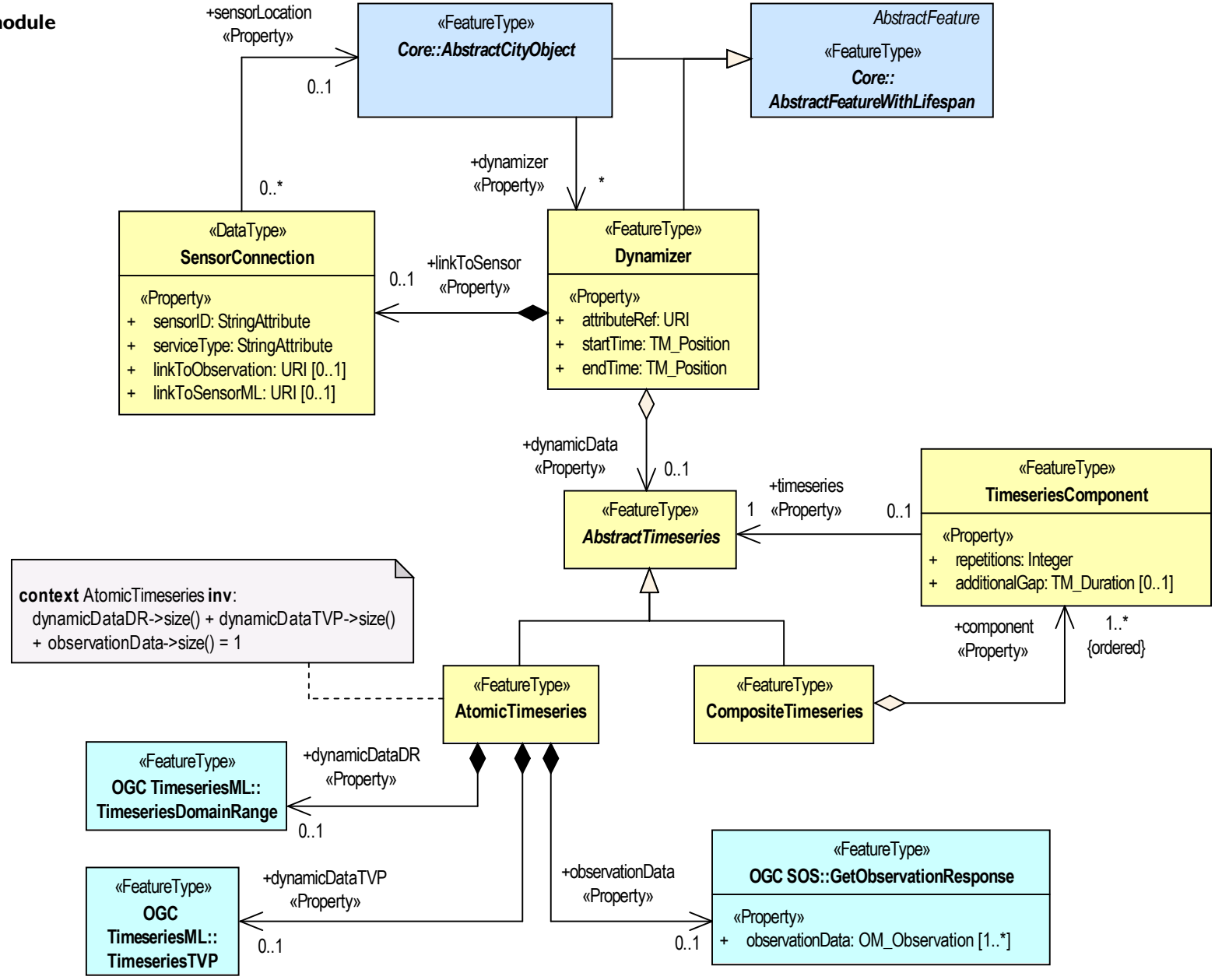
«enumeration»
ElevationReferenceValue

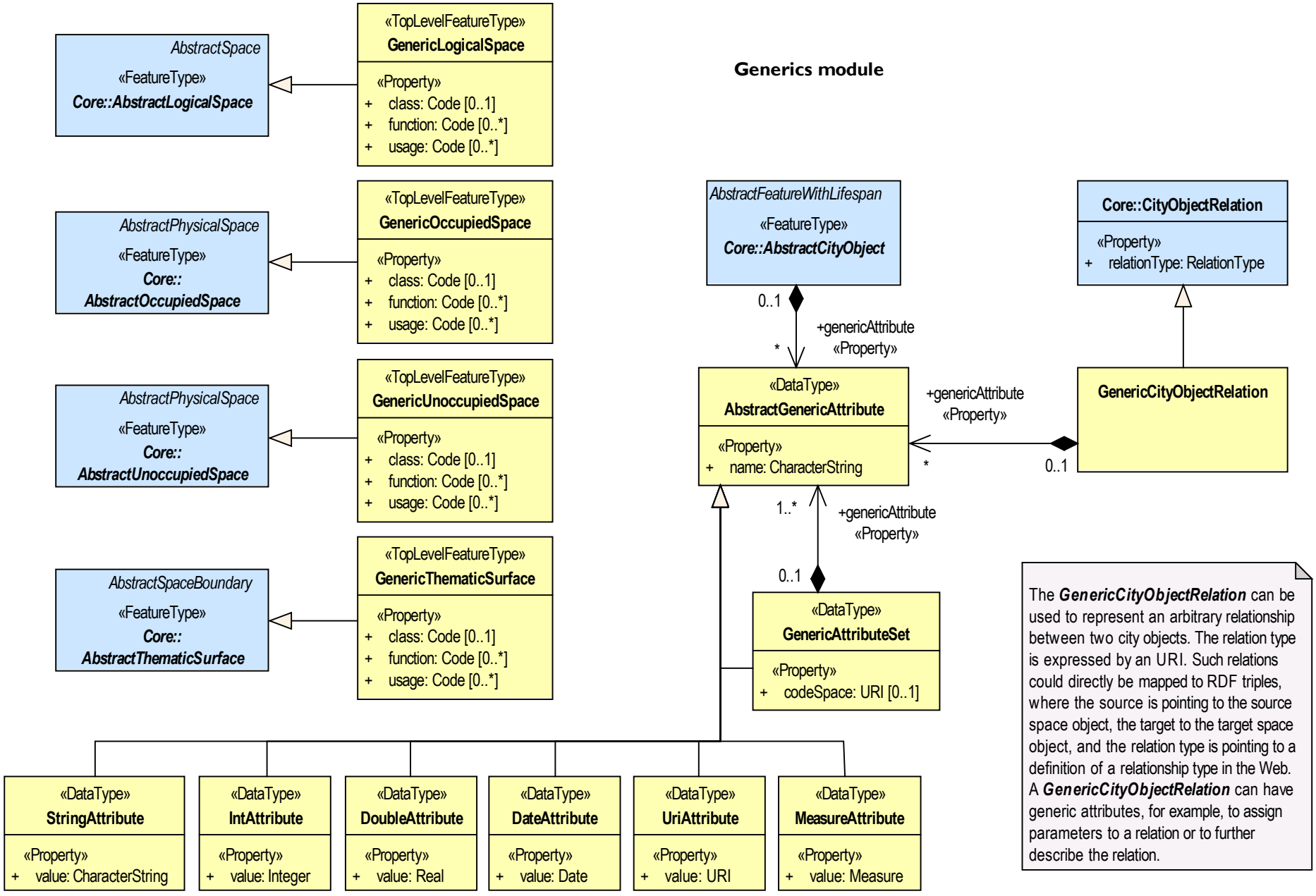
- aboveGroundEnvelope
- bottomOfConstruction
- entrancePoint
- generalEave
- generalGround
- generalRoof
- generalRoofEdge
- highestEave
- highestGroundPoint
- highestPoint
- highestRoofEdge
- lowestEave
- lowestFloorAboveGround
- lowestGroundPoint
- lowestRoofEdge
- topOfConstruction

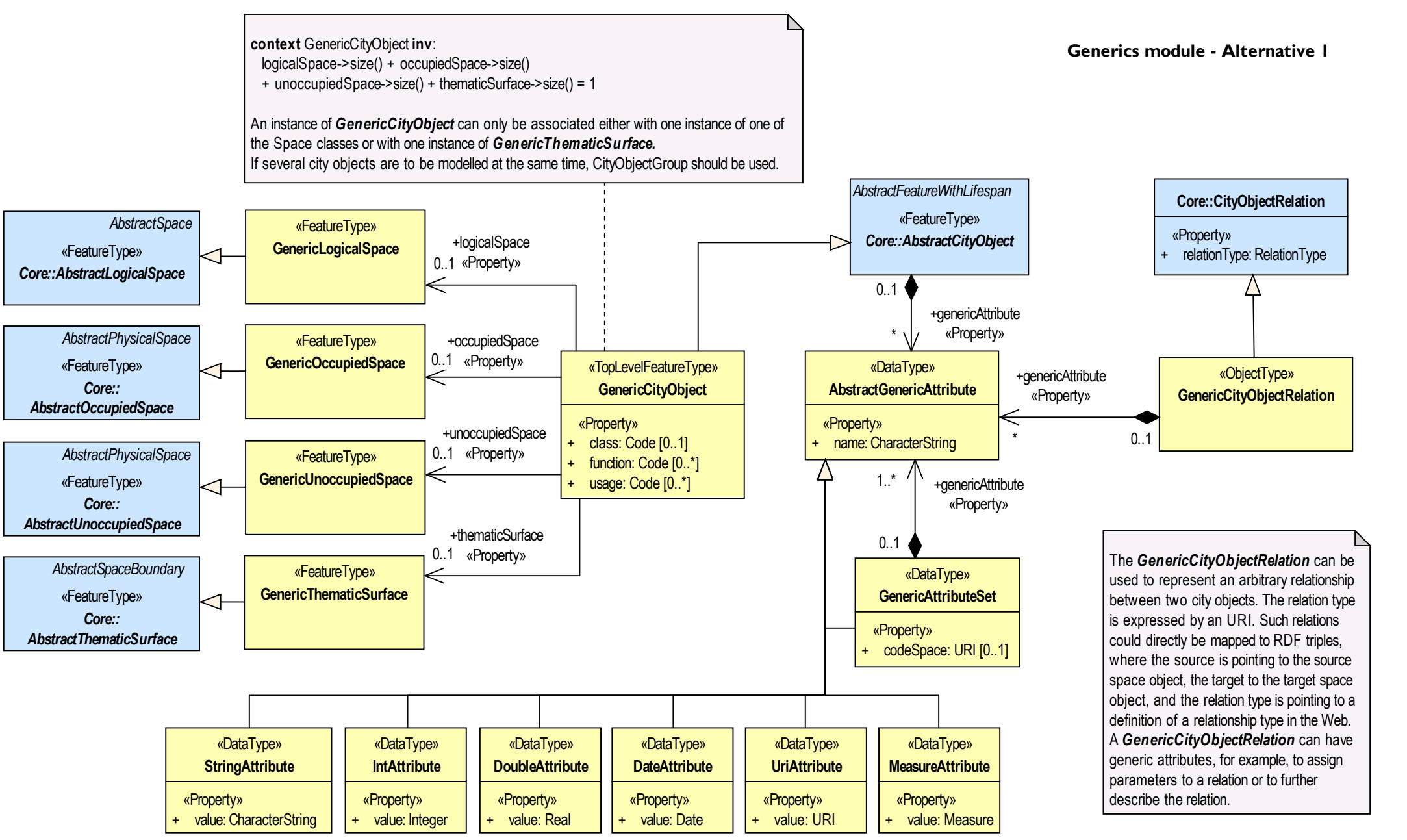
«enumeration»
HeightStatusValue

- estimated
- measured

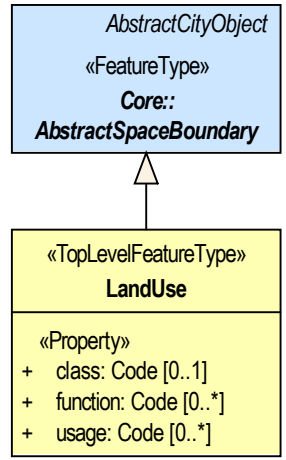
Dynamizer module



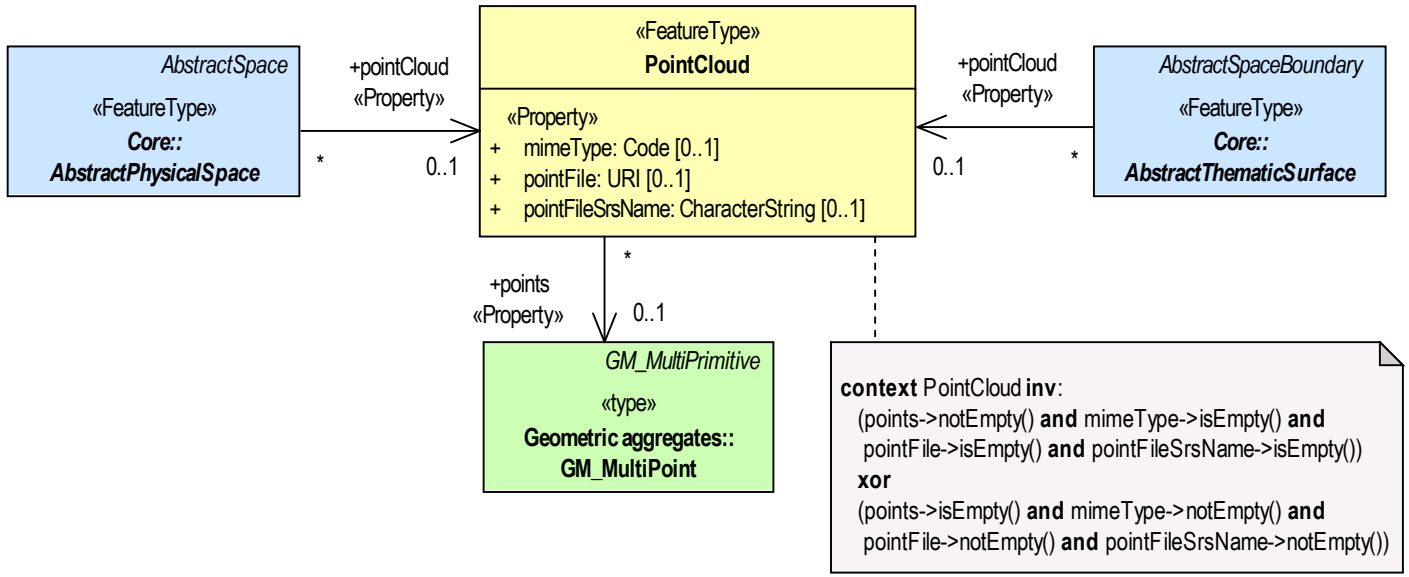

Generics module
AbstractSpace
«FeatureType»
Core::AbstractLogicalSpace
«TopLevelFeatureType»
GenericLogicalSpace
«Property»
+ class: Code [0..1]
+ function: Code [0..*]
+ usage: Code [0..*]
AbstractPhysicalSpace
«FeatureType»
Core::
AbstractOccupiedSpace
«TopLevelFeatureType»
GenericOccupiedSpace
«Property»
+ class: Code [0..1]
+ function: Code [0..*]
+ usage: Code [0..*]
AbstractPhysicalSpace
«FeatureType»
Core::
AbstractUnoccupiedSpace
«TopLevelFeatureType»
GenericUnoccupiedSpace
«Property»
+ class: Code [0..1]
+ function: Code [0..*]
+ usage: Code [0..*]
AbstractSpaceBoundary
«FeatureType»
Core::
AbstractThematicSurface
«TopLevelFeatureType»
GenericThematicSurface
«Property»
+ class: Code [0..1]
+ function: Code [0..*]
+ usage: Code [0..*]
AbstractFeatureWithLifespan
«FeatureType»
Core::AbstractCityObject
0..1
*
+genericAttribute
«Property»
«DataType»
AbstractGenericAttribute
«Property»
+ name: CharacterString
+genericAttribute
«Property»
*
0..1
Core::CityObjectRelation
«Property»
+ relationType: RelationType
GenericCityObjectRelation
1..*
+genericAttribute
«Property»
0..1
«DataType»
GenericAttributeSet
«Property»
+ codeSpace: URI [0..1]
«DataType»
StringAttribute
«Property»
+ value: CharacterString
«DataType»
IntAttribute
«Property»
+ value: Integer
«DataType»
DoubleAttribute
«Property»
+ value: Real
«DataType»
DateAttribute
«Property»
+ value: Date
«DataType»
UriAttribute
«Property»
+ value: URI
«DataType»
MeasureAttribute
«Property»
+ value: Measure
The GenericCityObjectRelation can be used to represent an arbitrary relationship between two city objects. The relation type is expressed by an URI. Such relations could directly be mapped to RDF triples, where the source is pointing to the source space object, the target to the target space object, and the relation type is pointing to a definition of a relationship type in the Web. A GenericCityObjectRelation can have generic attributes, for example, to assign parameters to a relation or to further describe the relation.

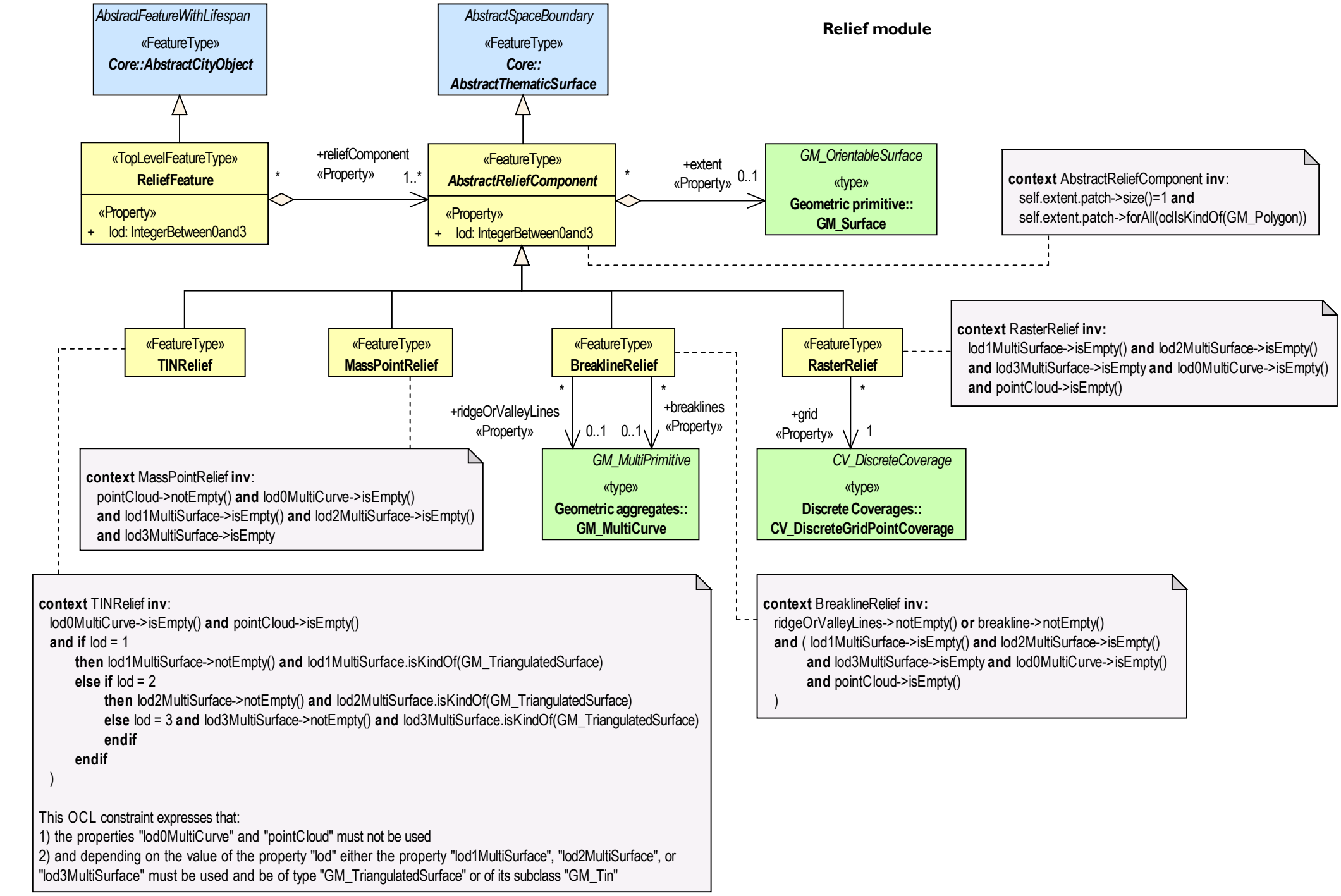

context GenericCityObject inv:
logicalSpace->size() + occupiedSpace->size()
+ unoccupiedSpace->size() + thematicSurface->size() = 1
An instance of GenericCityObject can only be associated either with one instance of one of the Space classes or with one instance of GenericThematicSurface.
If several city objects are to be modelled at the same time, CityObjectGroup should be used.
Generics module - Alternative I
AbstractSpace
«FeatureType»
Core::AbstractLogicalSpace
«FeatureType»
GenericLogicalSpace
+logicalSpace
0..1 «Property»
AbstractPhysicalSpace
«FeatureType»
Core::
AbstractOccupiedSpace
«FeatureType»
GenericOccupiedSpace
+occupiedSpace
0..1 «Property»
AbstractPhysicalSpace
«FeatureType»
Core::
AbstractUnoccupiedSpace
«FeatureType»
GenericUnoccupiedSpace
+unoccupiedSpace
0..1 «Property»
AbstractSpaceBoundary
«FeatureType»
Core::
AbstractThematicSurface
«FeatureType»
GenericThematicSurface
+thematicSurface
0..1 «Property»
«TopLevelFeatureType»
GenericCityObject
«Property»
+ class: Code [0..1]
+ function: Code [0..*]
+ usage: Code [0..*]
AbstractFeatureWithLifespan
«FeatureType»
Core::AbstractCityObject
0..1
*
+genericAttribute
«Property»
«DataType»
AbstractGenericAttribute
«Property»
+ name: CharacterString
+genericAttribute
«Property»
*
1..*
0..1
+genericAttribute
«Property»
«DataType»
GenericAttributeSet
«Property»
+ codeSpace: URI [0..1]
Core::CityObjectRelation
«Property»
+ relationType: RelationType
«ObjectType»
GenericCityObjectRelation
0..1
«DataType»
StringAttribute
«Property»
+ value: CharacterString
«DataType»
IntAttribute
«Property»
+ value: Integer
«DataType»
DoubleAttribute
«Property»
+ value: Real
«DataType»
DateAttribute
«Property»
+ value: Date
«DataType»
UriAttribute
«Property»
+ value: URI
«DataType»
MeasureAttribute
«Property»
+ value: Measure
The GenericCityObjectRelation can be used to represent an arbitrary relationship between two city objects. The relation type is expressed by an URI. Such relations could directly be mapped to RDF triples, where the source is pointing to the source space object, the target to the target space object, and the relation type is pointing to a definition of a relationship type in the Web. A GenericCityObjectRelation can have generic attributes, for example, to assign parameters to a relation or to further describe the relation.

LandUse module



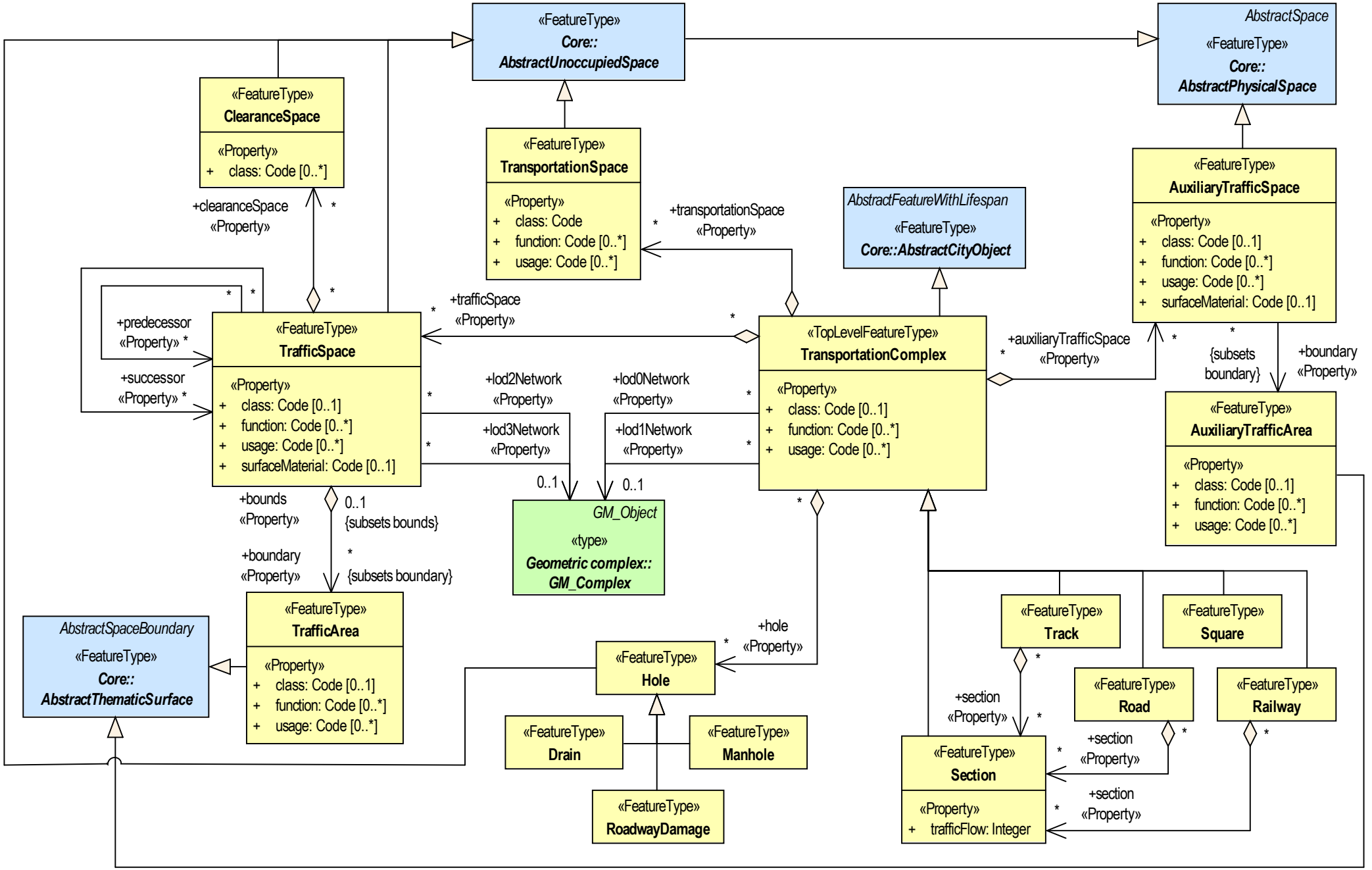
PointCloud module



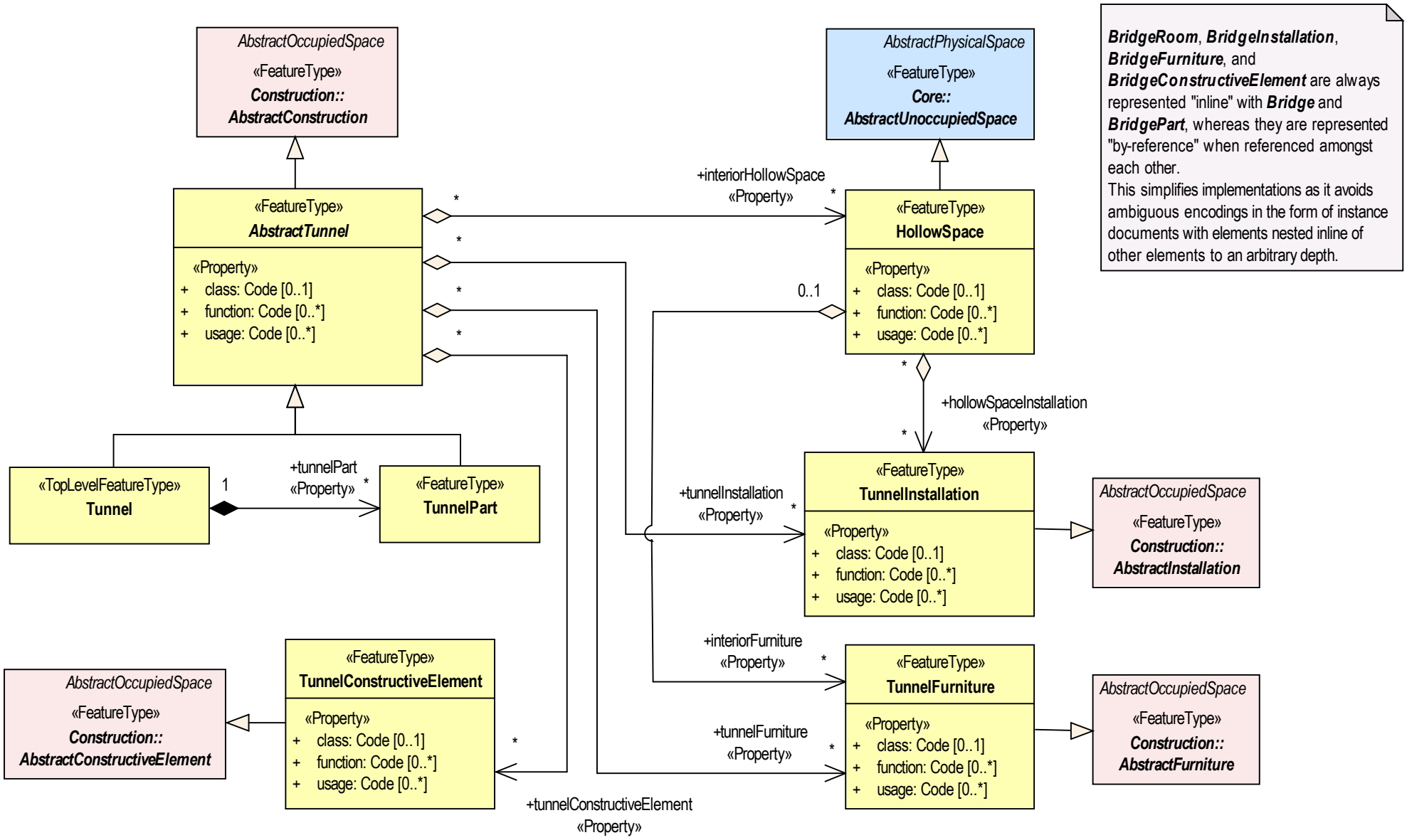

Relief module
AbstractFeatureWithLifespan
«FeatureType»
Core::AbstractCityObject
AbstractSpaceBoundary
«FeatureType»
Core::
AbstractThematicSurface
«TopLevelFeatureType»
ReliefFeature
«Property»
+ lod: IntegerBetween0and3
+reliefComponent
«Property»
1..*
«FeatureType»
AbstractReliefComponent
«Property»
+ lod: IntegerBetween0and3
+extent
«Property»
0..1
GM_OrientableSurface
«type»
Geometric primitive::
GM_Surface
context AbstractReliefComponent inv:
self.extent.patch->size()=1 and
self.extent.patch->forAll(oclIsKindOf(GM_Polygon))
«FeatureType»
TINRelief
«FeatureType»
MassPointRelief
«FeatureType»
BreaklineRelief
«FeatureType»
RasterRelief
context RasterRelief inv:
lod1MultiSurface->isEmpty() and lod2MultiSurface->isEmpty()
and lod3MultiSurface->isEmpty and lod0MultiCurve->isEmpty()
and pointCloud->isEmpty()
+ridgeOrValleyLines
«Property»
0..1
0..1
+breaklines
«Property»
+grid
«Property»
1
GM_MultiPrimitive
«type»
Geometric aggregates::
GM_MultiCurve
CV_DiscreteCoverage
«type»
Discrete Coverages::
CV_DiscreteGridPointCoverage
context MassPointRelief inv:
pointCloud->notEmpty() and lod0MultiCurve->isEmpty()
and lod1MultiSurface->isEmpty() and lod2MultiSurface->isEmpty()
and lod3MultiSurface->isEmpty
context TINRelief inv:
lod0MultiCurve->isEmpty() and pointCloud->isEmpty()
and if lod = 1
then lod1MultiSurface->notEmpty() and lod1MultiSurface.isKindOf(GM_TriangulatedSurface)
else if lod = 2
then lod2MultiSurface->notEmpty() and lod2MultiSurface.isKindOf(GM_TriangulatedSurface)
else lod = 3 and lod3MultiSurface->notEmpty() and lod3MultiSurface.isKindOf(GM_TriangulatedSurface)
endif
endif
)
This OCL constraint expresses that:
1) the properties "lod0MultiCurve" and "pointCloud" must not be used
2) and depending on the value of the property "lod" either the property "lod1MultiSurface", "lod2MultiSurface", or
"lod3MultiSurface" must be used and be of type "GM_TriangulatedSurface" or of its subclass "GM_Tin"
context BreaklineRelief inv:
ridgeOrValleyLines->notEmpty() or breakline->notEmpty()
and (lod1MultiSurface->isEmpty() and lod2MultiSurface->isEmpty()
and lod3MultiSurface->isEmpty and lod0MultiCurve->isEmpty()
and pointCloud->isEmpty()
)

Transportation module

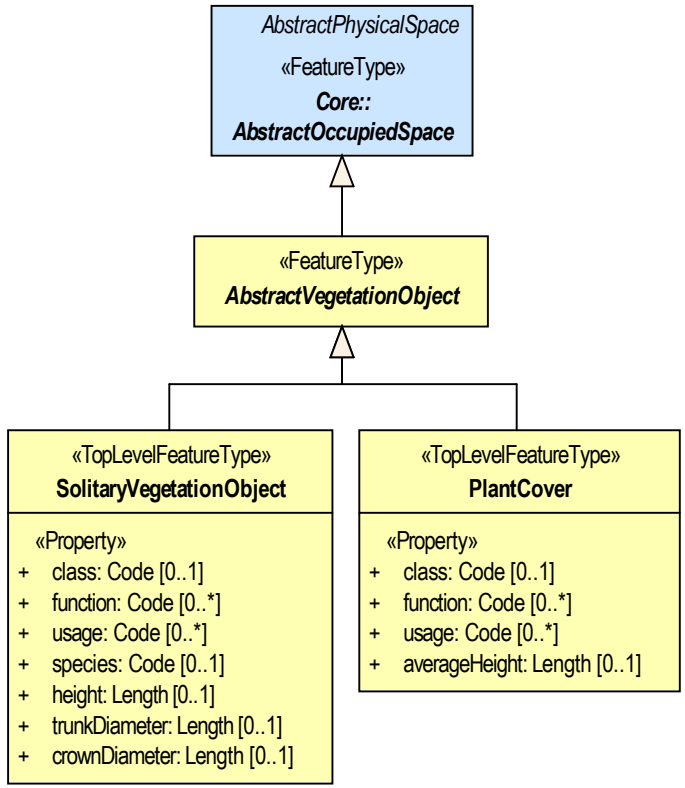
At the 3DGeoInfo conference at TU Delft early October 2018, a discussion between Anna Labetski and Thomas H. Kolbe took place regarding some refinements proposed by TU Delft in a conference paper.
These refinements are going to be integrated. The current diagram only reflects the refined modelling of the TopLevelFeatureType concept.



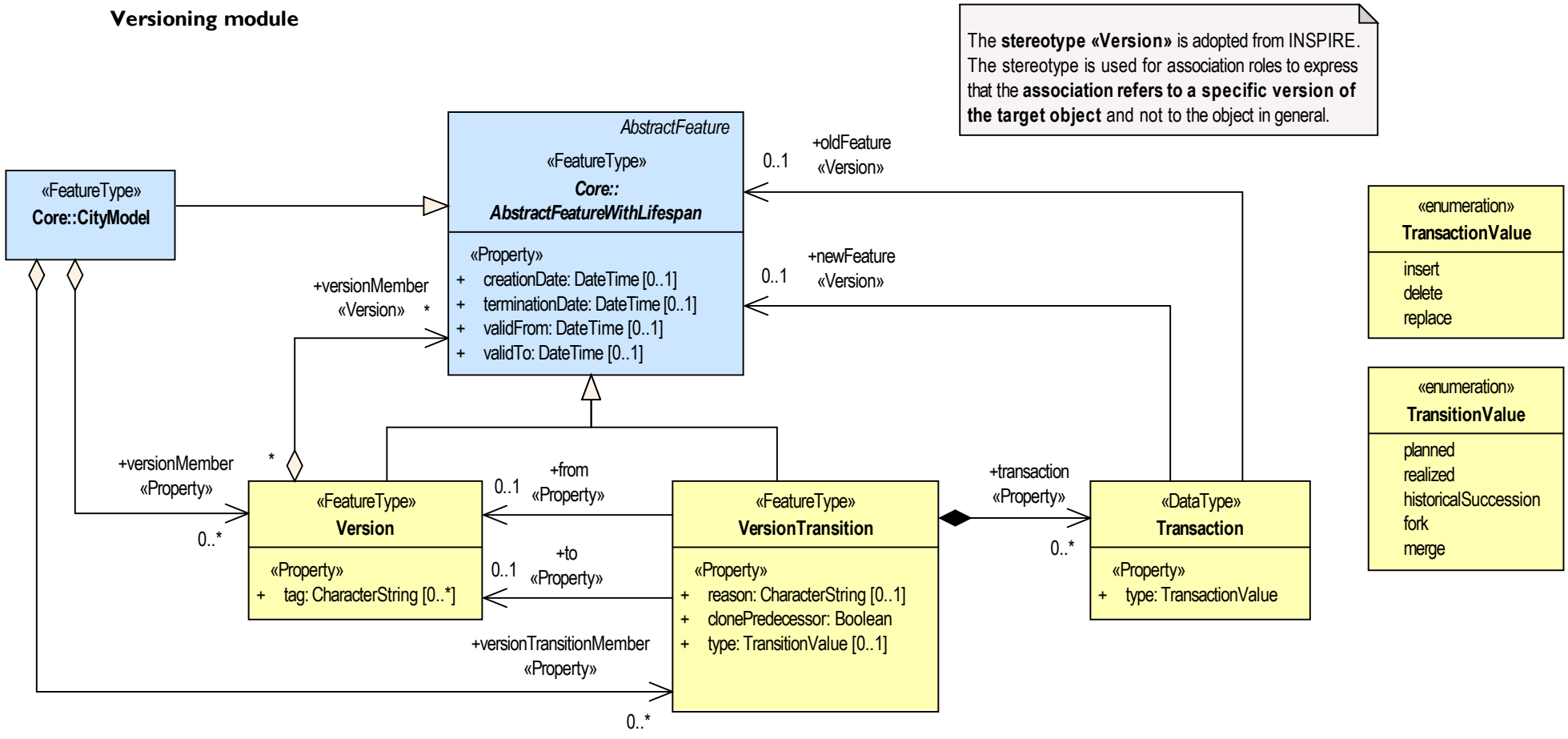
Tunnel module



Vegetation module



Versioning module



WaterBody module