

Open Geospatial Consortium

Submission Date: <yyyy-mm-dd>

Approval Date: <yyyy-mm-dd>

Publication Date: <2020-01-30>

External identifier of this OGC® document: <http://www.opengis.net/doc/IS/CityGML/3.0>

Internal reference number of this OGC® document: YY-nnnrx

Version: 0.4

Category: OGC® Conceptual Model

Editor: Charles Heazel

OGC City Geography Markup Language (CityGML) Conceptual Model Standard

Copyright notice

Copyright © 2019 Open Geospatial Consortium

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>

Warning

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OGC® Standard

Document subtype: Conceptual Model

Document stage: Draft

Document language: English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

Table of Contents

1. Introduction	19
1.1. Motivation	19
1.2. Historical background	19
1.3. Additions in CityGML 2.0	21
2. Scope	24
3. Conformance	26
4. References	27
5. Terms and Definitions	29
5.1. term name	29
5.2. Abbreviated Terms	29
6. Conventions	31
6.1. Identifiers	31
6.2. UML Notation	31
6.3. XML namespaces and namespace prefixes	33
7. Overview of CityGML	36
8. General characteristics of CityGML	37
8.1. Modularisation	37
8.2. Multi-scale modelling (5 levels of detail, LOD)	37
8.3. Coherent semantical-geometrical modelling	39
8.4. Closure surfaces	40
8.5. Terrain Intersection Curve (TIC)	41
8.6. Code lists for enumerative attributes	43
8.7. External references	44
8.8. City object groups	45
8.9. Appearances	45
8.10. Prototypic objects / scene graph concepts	46
8.11. Generic city objects and attributes	47
8.12. Application Domain Extensions (ADE)	47
9. Modularization	49
9.1. CityGML core and extension modules	50
9.2. CityGML profiles	56
10. Spatial Model	58
10.1. Geometric-topological model	58
10.1.1. Primitives and Composites	58
10.1.2. Complexes and Aggregates	59
10.1.3. Combined Geometries	59
10.1.4. Recursive Aggregation	60
10.2. Spatial Reference System	61

10.3. Implicit geometries, prototypic objects, scene graph concepts	61
11. CityGML Conceptual Model	62
12. Appearance Model	63
12.1. Appearance Model Data Dictionary	65
12.1.1. Class AbstractSurfaceData	65
12.1.2. Class AbstractTexture	66
12.1.3. Class Appearance	68
12.1.4. Class Color	69
12.1.5. Class ColorPlusOpacity	70
12.1.6. Class GeoreferencedTexture	71
12.1.7. Class ParameterizedTexture	72
12.1.8. Class TextureAssociation	73
12.1.9. Class X3DMaterial	74
12.1.10. Class AbstractTextureParameterization	76
12.1.11. Class TexCoordGen	78
12.1.12. Class TexCoordList	79
12.1.13. Class TextureType	80
12.1.14. Class WrapMode	81
12.1.15. Additional Information	82
12.2. Core	82
12.3. Core Model Data Dictionary	84
12.3.1. Class AbstractAppearance	84
12.3.2. Class AbstractCityObject	85
12.3.3. Class AbstractDynamizer	89
12.3.4. Class AbstractFeatureWithLifespan	90
12.3.5. Class AbstractLogicalSpace	93
12.3.6. Class AbstractOccupiedSpace	94
12.3.7. Class AbstractPhysicalSpace	97
12.3.8. Class AbstractPointCloud	99
12.3.9. Class AbstractSpace	100
12.3.10. Class AbstractSpaceBoundary	104
12.3.11. Class AbstractThematicSurface	105
12.3.12. Class AbstractUnoccupiedSpace	110
12.3.13. Class AbstractVersion	112
12.3.14. Class AbstractVersionTransition	114
12.3.15. Class Address	115
12.3.16. Class CityModel	117
12.3.17. Class CityObjectRelation	118
12.3.18. Class ClosureSurface	119
12.3.19. Class DoubleBetween0and1	120
12.3.20. Class DoubleBetween0and1List	121

12.3.21. Class DoubleList	122
12.3.22. Class ImplicitGeometry	124
12.3.23. Class IntegerBetween0and3	126
12.3.24. Class IntervalValue	127
12.3.25. Class MimeValue	127
12.3.26. Class OccupantTypeValue	128
12.3.27. Class OtherRelationTypeValue	128
12.3.28. Class QualifiedAreaValue	129
12.3.29. Class QualifiedVolumeValue	129
12.3.30. Class RelationTypeValue	130
12.3.31. Class TemporalRelationTypeValue	131
12.3.32. Class TopologicRelationTypeValue	132
12.3.33. Class TransformationMatrix2x2	132
12.3.34. Class TransformationMatrix3x4	133
12.3.35. Class TransformationMatrix4x4	134
12.3.36. Class AbstractGenericAttribute	135
12.3.37. Class ExternalReference	138
12.3.38. Class Occupancy	139
12.3.39. Class QualifiedArea	140
12.3.40. Class QualifiedVolume	141
12.3.41. Class RelativeToTerrain	142
12.3.42. Class RelativeToWater	143
12.3.43. Class SpaceType	144
12.3.44. Class XALAddressDetails	145
12.3.45. Additional Information	146
12.4. Digital Terrain Model	146
12.5. Relief Model Data Dictionary	147
 12.5.1. Class AbstractReliefComponent	147
 12.5.2. Class BreaklineRelief	150
 12.5.3. Class MassPointRelief	151
 12.5.4. Class RasterRelief	152
 12.5.5. Class ReliefFeature	153
 12.5.6. Class TINRelief	154
 12.5.7. Additional Information	155
12.6. Building Model	155
12.7. Building Model Data Dictionary	156
 12.7.1. Class AbstractBuilding	156
 12.7.2. Class AbstractBuildingSubdivision	160
 12.7.3. Class Building	162
 12.7.4. Class BuildingClassValue	163
 12.7.5. Class BuildingConstructiveElement	164

12.7.6. Class BuildingConstructiveElementClassValue	165
12.7.7. Class BuildingConstructiveElementFunctionValue	166
12.7.8. Class BuildingConstructiveElementUsageValue	166
12.7.9. Class BuildingFunctionValue	166
12.7.10. Class BuildingFurniture	167
12.7.11. Class BuildingFurnitureClassValue	169
12.7.12. Class BuildingFurnitureFunctionValue	169
12.7.13. Class BuildingFurnitureUsageValue	170
12.7.14. Class BuildingInstallation	170
12.7.15. Class BuildingInstallationClassValue	172
12.7.16. Class BuildingInstallationFunctionValue	172
12.7.17. Class BuildingInstallationUsageValue	173
12.7.18. Class BuildingPart	173
12.7.19. Class BuildingRoom	174
12.7.20. Class BuildingRoomClassValue	176
12.7.21. Class BuildingRoomFunctionValue	177
12.7.22. Class BuildingRoomUsageValue	177
12.7.23. Class BuildingSubdivisionClassValue	178
12.7.24. Class BuildingSubdivisionFunctionValue	178
12.7.25. Class BuildingSubdivisionUsageValue	179
12.7.26. Class BuildingUnit	179
12.7.27. Class BuildingUsageValue	180
12.7.28. Class RoofTypeValue	181
12.7.29. Class RoomElevationReferenceValue	181
12.7.30. Class Storey	181
12.7.31. Class RoomHeight	183
12.7.32. Additional Information	184
12.7.33. Boundary surfaces	186
12.7.34. Openings	187
12.7.35. Building Interior	188
12.7.36. Modelling building storeys using CityObjectGroups	188
12.8. Tunnel	189
12.9. Tunnel Model Data Dictionary	190
12.9.1. Class AbstractTunnel	190
12.9.2. Class HollowSpace	192
12.9.3. Class HollowSpaceClassValue	194
12.9.4. Class HollowSpaceFunctionValue	195
12.9.5. Class HollowSpaceUsageValue	195
12.9.6. Class Tunnel	196
12.9.7. Class TunnelClassValue	197
12.9.8. Class TunnelConstructiveElement	197

12.9.9. Class TunnelConstructiveElementClassValue	198
12.9.10. Class TunnelConstructiveElementFunctionValue	199
12.9.11. Class TunnelConstructiveElementUsageValue	199
12.9.12. Class TunnelFunctionValue	200
12.9.13. Class TunnelFurniture	200
12.9.14. Class TunnelFurnitureClassValue	202
12.9.15. Class TunnelFurnitureFunctionValue	202
12.9.16. Class TunnelFurnitureUsageValue	203
12.9.17. Class TunnelInstallation	203
12.9.18. Class TunnelInstallationClassValue	205
12.9.19. Class TunnelInstallationFunctionValue	205
12.9.20. Class TunnelInstallationUsageValue	206
12.9.21. Class TunnelPart	206
12.9.22. Class TunnelUsageValue	207
12.9.23. Additional Information	208
12.10. Bridge Model	208
12.11. Bridge Model Data Dictionary	209
12.11.1. Class AbstractBridge	209
12.11.2. Class Bridge	212
12.11.3. Class BridgeClassValue	213
12.11.4. Class BridgeConstructiveElement	213
12.11.5. Class BridgeConstructiveElementClassValue	215
12.11.6. Class BridgeConstructiveElementFunctionValue	215
12.11.7. Class BridgeConstructiveElementUsageValue	216
12.11.8. Class BridgeFunctionValue	216
12.11.9. Class BridgeFurniture	217
12.11.10. Class BridgeFurnitureClassValue	218
12.11.11. Class BridgeFurnitureFunctionValue	219
12.11.12. Class BridgeFurnitureUsageValue	219
12.11.13. Class BridgeInstallation	220
12.11.14. Class BridgeInstallationClassValue	221
12.11.15. Class BridgeInstallationFunctionValue	222
12.11.16. Class BridgeInstallationUsageValue	222
12.11.17. Class BridgePart	223
12.11.18. Class BridgeRoom	224
12.11.19. Class BridgeRoomClassValue	225
12.11.20. Class BridgeRoomFunctionValue	226
12.11.21. Class BridgeRoomUsageValue	226
12.11.22. Class BridgeUsageValue	227
12.11.23. Additional Information	227
12.12. Water Body	227

12.13. WaterBody Model Data Dictionary	228
12.13.1. Class AbstractWaterBoundarySurface	228
12.13.2. Class WaterBody	230
12.13.3. Class WaterBodyClassValue	231
12.13.4. Class WaterBodyFunctionValue	232
12.13.5. Class WaterBodyUsageValue	232
12.13.6. Class WaterGroundSurface	232
12.13.7. Class WaterLevelValue	233
12.13.8. Class WaterSurface	234
12.13.9. Additional Information	235
12.14. Transportation	235
12.15. Transportation Model Data Dictionary	236
12.15.1. Class AbstractTransportationSpace	236
12.15.2. Class AuxiliaryTrafficArea	239
12.15.3. Class AuxiliaryTrafficAreaClassValue	241
12.15.4. Class AuxiliaryTrafficAreaFunctionValue	241
12.15.5. Class AuxiliaryTrafficAreaUsageValue	242
12.15.6. Class AuxiliaryTrafficSpace	242
12.15.7. Class AuxiliaryTrafficSpaceClassValue	244
12.15.8. Class AuxiliaryTrafficSpaceFunctionValue	245
12.15.9. Class AuxiliaryTrafficSpaceUsageValue	245
12.15.10. Class ClearanceSpace	245
12.15.11. Class ClearanceSpaceClassValue	247
12.15.12. Class Hole	247
12.15.13. Class HoleClassValue	248
12.15.14. Class HoleSurface	249
12.15.15. Class Intersection	250
12.15.16. Class IntersectionClassValue	251
12.15.17. Class Marking	252
12.15.18. Class MarkingClassValue	253
12.15.19. Class Railway	253
12.15.20. Class RailwayClassValue	255
12.15.21. Class RailwayFunctionValue	255
12.15.22. Class RailwayUsageValue	256
12.15.23. Class Road	256
12.15.24. Class RoadClassValue	257
12.15.25. Class RoadFunctionValue	258
12.15.26. Class RoadUsageValue	258
12.15.27. Class Section	259
12.15.28. Class SectionClassValue	260
12.15.29. Class Square	261

12.15.30. Class SquareClassValue	262
12.15.31. Class SquareFunctionValue	262
12.15.32. Class SquareUsageValue	263
12.15.33. Class SurfaceMaterialValue	263
12.15.34. Class Track	264
12.15.35. Class TrackClassValue	265
12.15.36. Class TrackFunctionValue	266
12.15.37. Class TrackUsageValue	266
12.15.38. Class TrafficArea	266
12.15.39. Class TrafficAreaClassValue	268
12.15.40. Class TrafficAreaFunctionValue	268
12.15.41. Class TrafficAreaUsageValue	269
12.15.42. Class TrafficSpace	269
12.15.43. Class TrafficSpaceClassValue	272
12.15.44. Class TrafficSpaceFunctionValue	273
12.15.45. Class TrafficSpaceUsageValue	273
12.15.46. Class TransportationSpaceClassValue	274
12.15.47. Class TransportationSpaceFunctionValue	274
12.15.48. Class TransportationSpaceUsageValue	274
12.15.49. Class Waterway	275
12.15.50. Class WaterwayClassValue	276
12.15.51. Class WaterwayFunctionValue	277
12.15.52. Class WaterwayUsageValue	277
12.15.53. Class GranularityValue	278
12.15.54. Class TrafficDirectionValue	278
12.15.55. Additional Information	279
12.16. Vegetation	279
12.17. Vegetation Model Data Dictionary	281
 12.17.1. Class AbstractVegetationObject	281
 12.17.2. Class PlantCover	282
 12.17.3. Class PlantCoverClassValue	284
 12.17.4. Class PlantCoverFunctionValue	284
 12.17.5. Class PlantCoverUsageValue	284
 12.17.6. Class SolitaryVegetationObject	285
 12.17.7. Class SolitaryVegetationObjectClassValue	287
 12.17.8. Class SolitaryVegetationObjectFunctionValue	287
 12.17.9. Class SolitaryVegetationObjectUsageValue	288
 12.17.10. Class SpeciesValue	288
 12.17.11. Additional Information	289
12.18. City Furniture Model	289
12.19. CityFurniture Model Data Dictionary	290

12.19.1. Class CityFurniture	290
12.19.2. Class CityFurnitureClassValue	292
12.19.3. Class CityFurnitureFunctionValue	292
12.19.4. Class CityFurnitureUsageValue	293
12.19.5. Additional Information	293
12.20. Land Use	293
12.21. LandUse Model Data Dictionary	294
12.21.1. Class LandUse	294
12.21.2. Class LandUseClassValue	296
12.21.3. Class LandUseFunctionValue	296
12.21.4. Class LandUseUsageValue	297
12.21.5. Additional Information	297
12.22. City Object Group	297
12.23. CityObjectGroup Model Data Dictionary	298
12.23.1. Class CityObjectGroup	298
12.23.2. Class CityObjectGroupClassValue	300
12.23.3. Class CityObjectGroupFunctionValue	300
12.23.4. Class CityObjectGroupUsageValue	301
12.23.5. Class Role	301
12.23.6. Additional Information	302
12.24. Generics	302
12.25. Generics Model Data Dictionary	303
12.25.1. Class GenericLogicalSpace	303
12.25.2. Class GenericLogicalSpaceClassValue	305
12.25.3. Class GenericLogicalSpaceFunctionValue	305
12.25.4. Class GenericLogicalSpaceUsageValue	306
12.25.5. Class GenericOccupiedSpace	306
12.25.6. Class GenericOccupiedSpaceClassValue	307
12.25.7. Class GenericOccupiedSpaceFunctionValue	308
12.25.8. Class GenericOccupiedSpaceUsageValue	308
12.25.9. Class GenericThematicSurface	309
12.25.10. Class GenericThematicSurfaceClassValue	310
12.25.11. Class GenericThematicSurfaceFunctionValue	310
12.25.12. Class GenericThematicSurfaceUsageValue	311
12.25.13. Class GenericUnoccupiedSpace	311
12.25.14. Class GenericUnoccupiedSpaceClassValue	312
12.25.15. Class GenericUnoccupiedSpaceFunctionValue	313
12.25.16. Class GenericUnoccupiedSpaceUsageValue	313
12.25.17. Class DateAttribute	314
12.25.18. Class DoubleAttribute	315
12.25.19. Class GenericAttributeSet	316

12.25.20. Class IntAttribute	317
12.25.21. Class MeasureAttribute	318
12.25.22. Class StringAttribute	319
12.25.23. Class UriAttribute	321
12.25.24. Additional Information	322
12.26. Construction	322
12.27. Construction Model Data Dictionary	323
12.27.1. Class AbstractConstruction	323
12.27.2. Class AbstractConstructionSurface	326
12.27.3. Class AbstractConstructiveElement	328
12.27.4. Class AbstractFillingElement	330
12.27.5. Class AbstractFillingSurface	332
12.27.6. Class AbstractFurniture	333
12.27.7. Class AbstractInstallation	334
12.27.8. Class CeilingSurface	336
12.27.9. Class Door	337
12.27.10. Class DoorClassValue	338
12.27.11. Class DoorFunctionValue	339
12.27.12. Class DoorSurface	339
12.27.13. Class DoorUsageValue	340
12.27.14. Class ElevationReferenceValue	341
12.27.15. Class EventValue	341
12.27.16. Class FloorSurface	342
12.27.17. Class GroundSurface	343
12.27.18. Class InteriorWallSurface	344
12.27.19. Class OtherConstruction	345
12.27.20. Class OtherConstructionClassValue	346
12.27.21. Class OtherConstructionFunctionValue	347
12.27.22. Class OtherConstructionUsageValue	347
12.27.23. Class OuterCeilingSurface	347
12.27.24. Class OuterFloorSurface	348
12.27.25. Class RoofSurface	349
12.27.26. Class WallSurface	350
12.27.27. Class Window	351
12.27.28. Class WindowClassValue	352
12.27.29. Class WindowFunctionValue	353
12.27.30. Class WindowSurface	353
12.27.31. Class WindowUsageValue	354
12.27.32. Class ConditionOfConstructionValue	355
12.27.33. Class ConstructionEvent	356
12.27.34. Class Elevation	357

12.27.35. Class Height	358
12.27.36. Class HeightStatusValue	359
12.27.37. Class RelationToConstruction	360
12.27.38. Additional Information	360
12.28. Dynamizer	361
12.29. Dynamizer Model Data Dictionary	362
12.29.1. Class AbstractAtomicTimeseries	362
12.29.2. Class AbstractTimeseries	363
12.29.3. Class AuthenticationValue	365
12.29.4. Class CompositeTimeseries	365
12.29.5. Class Dynamizer	366
12.29.6. Class GenericTimeseries	368
12.29.7. Class SensorConnectionValue	369
12.29.8. Class StandardFileTimeseries	370
12.29.9. Class StandardFieldValue	371
12.29.10. Class TabulatedFileTimeseries	372
12.29.11. Class TabulatedFieldValue	375
12.29.12. Class SensorConnection	375
12.29.13. Class TimeseriesComponent	378
12.29.14. Class TimeseriesValue	379
12.29.15. Class TimeValuePair	380
12.29.16. Additional Information	382
13. Media Types for any data encoding(s)	384
Annex A: Conformance Class Abstract Test Suite (Normative)	385
A.1. Conformance Class Appearance	385
A.2. Conformance Class Bridge	394
A.3. Conformance Class Building	403
A.4. Conformance Class CityFurniture	417
A.5. Conformance Class CityObjectGroup	418
A.6. Conformance Class Core	421
A.7. Conformance Class Dynamizer	445
A.8. Conformance Class Generics	453
A.9. Conformance Class LandUse	464
A.10. Conformance Class PointCloud	466
A.11. Conformance Class Relief	467
A.12. Conformance Class Transportation	471
A.13. Conformance Class Tunnel	493
A.14. Conformance Class Vegetation	502
A.15. Conformance Class Versioning	506
A.16. Conformance Class WaterBody	509
Annex B: Title ({Normative/Informative})	514

Annex C: Revision History	515
14. Changelog for CityGML 3.0	516
Annex D: Bibliography	517

i. Abstract

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.2.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

ii. Keywords

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

ogcdoc, OGC document, <tags separated by commas>

iii. Preface

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.

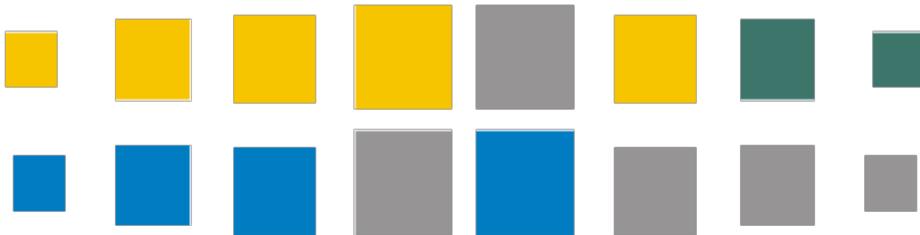
Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

There are significant changes between CityGML version 2.0.0 and CityGML version 1.0.0 (OGC document no. 08-007r1):

- New thematic modules for the representation of tunnels and bridges;
- Additional boundary surfaces for the semantic classification of the outer shell of buildings and building parts (OuterCeilingSurface and OuterFloorSurface);
- LOD0 representation (footprint and roof egde representations) for buildings and building parts;
- Additional attributes denoting a city object's location with respect to the surrounding terrain and water surface (relativeToTerrain and relativeToWater);
- Additional generic attributes for measured values and attribute sets; and
- Redesign of the CityGML code list mechanism (enumerative attributes are now of type `gml:CodeType` which facilitates to provide additional code lists enumerating their possible attribute values).

Migration of existing CityGML 1.0 instances to valid 2.0 instances only requires changing the CityGML namespace and schema location values in the document to the actual 2.0 values.

iv. Submitting organizations



CityGML

This is the official CityGML logo. For current news on CityGML and information about ongoing projects and fields of research in the area of CityGML see <http://www.citygml.org> and <http://www.citygmlwiki.org>

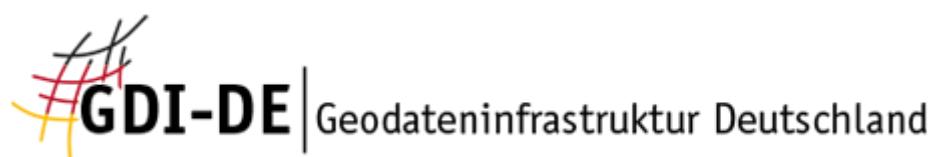
NOTE

PNG interlace method not supported for PDF generation. Find a more compatible OGC logo for "image::images/OGC_Logo.png[]"

OGC work on CityGML is discussed and coordinated by the OGC 3D Information Management (3DIM) Working Group. CityGML was initially implemented and evaluated as part of the OGC Web Services Testbed, Phase 4 (OWS-4) in the CAD/GIS/BIM thread.

Version 2.0 of this standards document was prepared by the OGC CityGML Standards Working Group (SWG). Future discussion and development will be led by the 3DIM Working Group.

For further information see <http://www.opengeospatial.org/projects/groups/3dimwg>



CityGML also continues to be developed by the members of the Special Interest Group 3D (SIG 3D) of the GDI-DE Geodateninfrastruktur Deutschland (Spatial Data Infrastructure Germany) in joint cooperation with the 3DIM Working Group and the CityGML SWG within OGC.

For further information see <http://www.sig3d.org/>



The preparation of the English document version and the European discussion has been supported by the European Spatial Data Research Organization (EuroSDR; formerly known as OEEPE) in an EuroSDR Commission III project. For further information see <http://www.eurosdr.net>

This Document was submitted to the Open Geospatial Consortium (OGC) by the members of the CityGML 3.0 Standards Working Group of the OGC. Amongst others, this comprises the following organizations:

- Autodesk, Inc. (primary submitter)
- Bentley Systems, Inc. (primary submitter)
- Technical University Berlin (submitter of technology)
- Ordnance Survey, UK
- University of Bonn, Germany
- Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam
- Institute for Applied Computer Science, Karlsruhe Institute of Technology

CityGML was originally developed by the Special Interest Group 3D (SIG 3D), 2002 – 2012 - www.citygml.org.

v. Submitters

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

Table 1. Submission Contact Points

Name	Institution	Email
Prof. Dr. Thomas H. Kolbe, Claus Nagel, Alexandra Lorenz	Institute for Geodesy and Geoinformation Science, Technical University Berlin	thomas.kolbe@tu-berlin.de claus.nagel@tu-berlin.de alexandra.lorenz@tu-berlin.de
Dr. Gerhard Gröger, Prof. Dr. Lutz Plümer, Angela Czerwinski	Institute for Geodesy and Geoinformation, University of Bonn	Groeger@ikg.uni-bonn.de Pluemer@ikg.uni-bonn.de Czerwinski@ikg.uni-bonn.de
Haik Lorenz	Autodesk, Inc.	haik.lorenz@autodesk.com
Alain Lapierre, Stefan Apfel, Paul Scarponcini	Bentley Systems, Inc.	alain.lapierre@bentley.com stefan.apfel@bentley.com paul.scarponcini@bentley.com
Carsten Rönsdorf	Ordnance Survey, Great Britain	carsten.roensdorf@ordnancesurvey.co.uk

Name	Institution	Email
Prof. Dr. Jürgen Döllner	Hasso-Plattner-Institute for IT Systems Engineering, University of Potsdam	juergen.doellner@hpi.uni-potsdam.de
Dr. Joachim Benner, Karl-Heinz Häfele	Institute for Applied Computer Science, Karlsruhe Institute of Technology	joachim.benner@kit.edu karl-heinz.haefele@kit.edu

vi. Participants in development

Table 2. Participants in Development

Name	Institution
Ulrich Gruber, Sandra Schlüter	District Administration Recklinghausen, Cadastre Department, Germany
Frank Bildstein	Rheinmetall Defence Electronics, Germany
Rüdiger Drees	T-Systems Enterprise Services GmbH, Bonn, Germany
Andreas Kohlhaas	GIStec GmbH (formerly), Germany
Frank Thiemann	Institute for Cartography and Geoinformatics, University of Hannover
Martin Degen	Cadastre Department, City of Dortmund
Heinrich Geerling	Architekturbüro Geerling, Germany
Dr. Frank Knospe	Cadastre and Mapping Department, City of Essen,
Hardo Müller	Snowflake Software Ltd., Great Britain
Martin Rechner	rechner logistic, Germany
Jörg Haist, Daniel Holweg	Fraunhofer Institute for Computer Graphics (IGD), Darmstadt, Germany
Prof. Dr. Peter A. Henning	Faculty for Computer Science, University of Applied Sciences, Karlsruhe, Germany
Rolf Wegener, Stephan Heitmann	State Cadastre and Mapping Agency of North-Rhine Westphalia, Germany
Prof. Dr. Marc-O. Löwner	Institute for Geodesy and Photogrammetry, Technical University of Braunschweig
Dr. Egbert Casper	Zerna Ingenieure, Germany
Christian Dahmen	con terra GmbH, Germany
Nobuhiro Ishimaru, Kishiko Maruyama, Eiichiro Umino, Takahiro Hirose	Hitachi, Ltd., Japan

Name	Institution
Linda van den Brink	Geonovum, The Netherlands
Ron Lake, David Burggraf	Galdos Systems Inc., Canada
Marie-Lise Vautier, Emmanuel Devys	Institut géographique national, France
Mark Pendlington	Ordnance Survey, Great Britain

vii. Acknowledgements

The SIG 3D wishes to thank the members of the CityGML Standards Working Group and the 3D Information Management (3DIM) Working Group of the OGC as well as all contributors of change requests and comments. In particular: Tim Case, Scott Simmons, Paul Cote, Clemens Portele, Jeffrey Bell, Chris Body, Greg Buehler, François Golay, John Herring, Jury Konga, Kai-Uwe Krause, Gavin Park, Richard Pearsall, George Percivall, Mauro Salvemini, Alessandro Triglia, David Wesloh, Tim Wilson, Greg Yetman, Jim Farley, Cliff Behrens, Lukas Herman, Danny Kita, and Simon Cox.

Further credits for careful reviewing and commenting of this document go to: Ludvig Emgard, Bettina Petzold, Dave Capstick, Mark Pendlington, Alain Lapierre, and Frank Steggink.

Chapter 1. Introduction

1.1. Motivation

An increasing number of cities and companies are building virtual 3D city models for different application areas like urban planning, mobile telecommunication, disaster management, 3D cadastre, tourism, vehicle and pedestrian navigation, facility management and environmental simulations. Furthermore, in the implementation of the European Environmental Noise Directive (END, 2002/49/EC) 3D geoinformation and 3D city models play an important role.

In recent years, most virtual 3D city models have been defined as purely graphical or geometrical models, neglecting the semantic and topological aspects. Thus, these models could almost only be used for visualisation purposes but not for thematic queries, analysis tasks, or spatial data mining. Since the limited reusability of models inhibits the broader use of 3D city models, a more general modelling approach had to be taken in order to satisfy the information needs of the various application fields.

CityGML is a common semantic information model for the representation of 3D urban objects that can be shared over different applications. The latter capability is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the possibility of selling the same data to customers from different application fields. The targeted application areas explicitly include city planning, architectural design, tourist and leisure activities, environmental simulation, mobile telecommunication, disaster management, homeland security, real estate management, vehicle and pedestrian navigation, and training simulators.

CityGML is designed as an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is implemented as an application schema of the Geography Markup Language 3 (GML3), the extendible international standard for spatial data exchange and encoding issued by the Open Geospatial Consortium (OGC) and the ISO TC211. CityGML is based on a number of standards from the ISO 191xx family, the Open Geospatial Consortium, the W3C Consortium, the Web 3D Consortium, and OASIS.

CityGML defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and appearance properties. “City” is broadly defined to comprise not just built structures, but also elevation, vegetation, water bodies, “city furniture”, and more. Included are generalisation hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. CityGML is applicable for large areas and small regions and can represent the terrain and 3D objects in different levels of detail simultaneously. Since either simple, single scale models without topology and few semantics or very complex multi-scale models with full topology and fine-grained semantical differentiations can be represented, CityGML enables lossless information exchange between different GI systems and users.

1.2. Historical background

CityGML has been developed since 2002 by the members of the Special Interest Group 3D (SIG 3D). Since 2010, this group is part of the initiative Spatial Data Infrastructure Germany (GDI-DE). Before

2010, the SIG 3D was affiliated to the initiative Geodata Infrastructure North Rhine-Westphalia (GDI NRW). The SIG 3D is an open group consisting of more than 70 companies, municipalities, and research institutions from Germany, Great Britain, Switzerland, and Austria working on the development and commercial exploitation of interoperable 3D city models and geovisualisation. Another result of the work from the SIG 3D is the proposition of the Web 3D Service (W3DS), a 3D portrayal service that is also being discussed in the Open Geospatial Consortium (OGC Doc. No. 05-019 and OGC Doc. No. 09-104r1).

A first successful implementation and evaluation of a subset of CityGML has been performed in the project “Pilot 3D” of the GDI NRW in 2005. Participants came from all over Germany and demonstrated city planning scenarios and tourist applications. By the beginning of 2006, a CityGML project within EuroSDR (European Spatial Data Research) started focusing on the European harmonisation of 3D city modelling. From June to December 2006, CityGML was employed and evaluated in the CAD/GIS/BIM thread of the OpenGIS Web Services Testbed #4 (OWS-4). Since 2008, CityGML (version 1.0.0) is an adopted OGC standard.

From that point in time, CityGML has disseminated worldwide. Many cities in Germany and in other countries in Europe provide their 3D city model in CityGML (Berlin, Cologne, Dresden and Munich, to mention only a few). In France, the project Bâti3D (IGN France) defines a profile of CityGML LOD2 and provides data from Paris and the city centres of Aix-en-Provence, Lille, Nantes and Marseille. CityGML also plays an important role in the pilot 3D project to obtain a 3D geoinformation standard and a 3D infrastructure for The Netherlands. Many cities in Europe like Monaco, Geneva, Zurich, Leewarden use CityGML LOD 2 or 3 to represent and exchange data, as well as cities in Denmark (LOD 2 and 3, partly LOD4). CityGML has strongly influenced the building model (version 2.0) of the INSPIRE initiative of the EU commission, which aims at the creation of an European spatial data infrastructure providing public sector data in an interoperable way. In Asia, the 3D city models of Istanbul (LOD 1 and 2), Doha, Katar (LOD3), and Yokohama (LOD2) are represented and exchanged in CityGML. Moreover, CityGML plays a crucial role for the 3D Spatial data infrastructure in Malaysia.

Today many commercial and academic tools support CityGML by providing import interfaces, export interfaces or both. An example is the 3D City Database which is a free and open source 3D geo database to store, represent, and manage virtual 3D city models on top of Oracle 10g R2 and 11g R1/R2 provided by the Technische Universität Berlin. It fully supports CityGML and is shipped with a tool for the import and export of CityGML models. Furthermore, an open source Java class library and API for the processing of CityGML models (citygml4j) is provided by the Technische Universität Berlin. The conversion tool FME (Feature Manipulation Engine) from Safe Software Inc., which is part of the interoperability extension of ESRI’s ArcGIS, has read and write interfaces for CityGML. The same applies to CAD tools as BentleyMap from Bentley Systems as well as to GIS tools like SupportGIS from CPA Geo-Information. Many 3D viewers (which all are freely available) provide read interfaces for CityGML: the Aristoteles Viewer from the University of Bonn, LandXplorer CityGML Viewer from Autodesk Inc. (the studio version for authoring and management is not free) and the FZKViewer for IFC and CityGML from KIT Karlsruhe and BS Contact from Bitmanagement Software GmbH which offers a CityGML plugin for the geospatial extension BS Contact Geo. This enumeration of software tools is not exhaustive and steadily growing. Please refer to the official website of CityGML at <http://www.citygml.org> as well as the CityGML Wiki at <http://www.citygmlwiki.org> for more information.

1.3. Additions in CityGML 2.0

CityGML 2.0 is a major revision of the previous version 1.0 of this International Standard (OGC Doc. No. 08-007r1), and introduces substantial additions and new features to the thematic model of CityGML. The revision was originally planned to be a minor update to version 1.1. The main endeavor of the revision process was to ensure backwards compatibility both on the level of the conceptual model and on the level of CityGML instance documents. However, some changes could not be implemented consistent with directives for minor revisions and backwards compatibility as enforced by OGC policy (cf. OGC Doc. No. 135r11). The major version number change to 2.0 is therefore a consequence of conforming to the OGC versioning policy without having to abandon any changes or additions which reflect requests from the CityGML community.

CityGML 2.0 is backwards compatible with version 1.0 in the following sense: each valid 1.0 instance is a valid 2.0 instance provided that the CityGML namespaces and schema locations in the document are changed to their actual 2.0 values. This step is required because the CityGML version number is encoded in these values, but no further actions have to be taken. Hence, there is a simple migration path from existing CityGML 1.0 instances to valid 2.0 instances.

The following clauses provide an overview of what is new in CityGML 2.0.

New thematic modules for the representation of bridges and tunnels

Bridges and tunnels are important objects in city and landscape models. They are an essential part of the trans-portation infrastructure and are often easily recognizable landmarks of a city. CityGML 1.0 has been lacking thematic modules dedicated to bridges and tunnels, and thus such objects had to be modelled and exchanged using a GenericCityObject as proxy (cf. chapter 10.12). CityGML 2.0 now introduces two new thematic modules for the explicit representation of bridges and tunnels which complement the thematic model of CityGML: the Bridge module (cf. chapter 10.4) and the Tunnel module (cf. chapter 10.5).

Bridges and tunnels can be represented in LOD 1 – 4 and the underlying data models have a coherent structure with the Building model. For example, bridges and tunnels can be decomposed into parts, thematic boundary surfaces with openings are available to semantically classify parts of the shell, and installations as well as interi-or built structures can be represented. This coherent model structure facilitates the similar understanding of semantic entities and helps to reduce software implementation efforts. Both the Bridge and the Tunnel model introduce further concepts and model elements which are specific to bridges and tunnels respectively.

Additions to existing thematic modules

- *CityGML Core module* (cf. chapter 10.1) Two new optional attributes have been added to the abstract base class *core:_CityObject* within the *CityGML Core* module: *relativeToTerrain* and *relativeToWater*. These attributes denote the feature's location with respect to the terrain and water surface in a qualitative way, and thus facilitate simple and efficient queries (e.g., for the number of subsurface buildings) without the need for an additional digital terrain model or a model of the water body.
- *Building module* (cf. chapter 10.3)
 - *LOD0 representation* : Buildings can now be represented in LOD0 by footprint and/or roof

edge polygons. This allows the easy integration of existing 2D data and of roof reconstructions from aerial and satellite imagery into a 3D city model. The representations are restricted to horizontal, 3-dimensional surfaces.

- *Additional thematic boundary surfaces* : In order to semantically classify parts of the outer building shell which are neither horizontal wall surfaces nor parts of the roof, two additional boundary surfaces are introduced: *OuterFloorSurface* and *OuterCeilingSurface*.
- *Additional relations to thematic boundary surfaces* : In addition to *_AbstractBuilding* and *Room*, the surface geometries of *BuildingInstallation* and *IntBuildingInstallation* features can now be semantically classified using thematic boundary surfaces. For example, this facilitates the semantic differentiation between roof and wall surfaces of dormers which are modeled as *BuildingInstallation*.
- *Additional use of implicit geometries* : Implicit geometries (cf. chapter 8.3) are now available for the representation of *_Opening*, *BuildingInstallation*, and *IntBuildingInstallation* in addition to *BuildingFurniture*. A prototypical geometry for these city objects can thus be stored once and instantiated at different locations in the 3D city model.
- *Generics module* (cf. chapter 10.12) Two generic attributes have been added to the *Generics* module: *MeasureAttribute* and *GenericAttributeSet*. A *MeasureAttribute* facilitates the representation of measured values together with a reference to the employed unit. A *GenericAttributeSet* is a named collection of arbitrary generic attributes. It provides an optional *codeSpace* attribute to denote the authority organization who defined the attribute set.
- *LandUse module* (cf. chapter 10.10) The scope of the feature type *LandUse* has been broadened to comprise both areas of the earth's surface dedicated to a specific land use and areas of the earth's surface having a specific land cover with or without vegetation.
- *Attributes class, function, and usage (all modules)* In order to harmonize the use of the attributes *class*, *function*, and *usage*, this attribute triplet has been complemented for all feature classes that at least provided one of the attributes in CityGML 1.0.

Additions to the CityGML code list mechanism

In CityGML, code lists providing the allowed values for enumerative attributes such as *class*, *function*, and *usage* can be specified outside the CityGML schema by any organization or information community according to their specific information needs. This mechanism is, however, not fully reflected in the CityGML 1.0 encoding schema, because in a CityGML 1.0 instance document a corresponding attribute cannot point to the dictionary with the used code list values. This has been corrected for CityGML 2.0: All attributes taking values from code lists are now of type *gml:CodeType* following the GML 3.1.1 mechanism for the encoding of code list values (cf. chapter 10.14 for more information). The *gml:CodeType* adds an optional *codeSpace* value to enumerative attributes which allows for providing a persistent URI pointing to the corresponding dictionary.

Changelog for CityGML 2.0

Changes on the level of XML schema components are provided in Annex F.

Further edits to the specification document

- *Accuracy requirements for Levels of Detail (LOD)* (cf. chapter 6.2) The accuracy requirements for the different CityGML LODs proposed in chapter 6.2 are non-normative. The wording of chapter

6.2 in CityGML 1.0 is however inconsistent with regard to this fact and thus has been clarified for CityGML 2.0.

- *Rework of the CityGML example datasets (cf. Annex G)* The CityGML examples provided in Annex G have been reworked and extended. They now show a consistent building model in all five LODs and demonstrate, for example, the semantic and geometric refinement of the building throughout the different LODs as well as the usage of XLinks to share geometry elements between features. The datasets are shipped with the CityGML XML Schema package, and are available at <http://schemas.opengis.net/citygml/examples/2.0/>.
- *New example for the usage of Application Domain Extensions (cf. Annex I)* A second example for the usage of Application Domain Extensions in the field of Ubiquitous Network Robots Services has been added in Annex I.

Chapter 2. Scope

This document is an OGC Encoding Standard for the representation, storage and exchange of virtual 3D city and landscape models. CityGML is implemented as an application schema of the Geography Markup Language version 3.1.1 (GML3).

CityGML models both complex and georeferenced 3D vector data along with the semantics associated with the data. In contrast to other 3D vector formats, CityGML is based on a rich, general purpose information model in addition to geometry and appearance information. For specific domain areas, CityGML also provides an extension mechanism to enrich the data with identifiable features under preservation of semantic interoperability.

Targeted application areas explicitly include urban and landscape planning; architectural design; tourist and leisure activities; 3D cadastres; environmental simulations; mobile telecommunications; disaster management; homeland security; vehicle and pedestrian navigation; training simulators and mobile robotics.

CityGML is considered a source format for 3D portraying. The semantic information contained in the model can be used in the styling process which generates computer graphics represented e.g. as KML/COLLADA or X3D files. The appropriate OGC Portrayal Web Service for this process is the OGC Web 3D Service (W3DS). An image-based 3D portrayal service for virtual 3D landscape and city models is provided by the OGC Web View Service (WVS).

Features of CityGML:

- Geospatial information model (ontology) for urban landscapes based on the ISO 191xx family
- GML3 representation of 3D geometries, based on the ISO 19107 model
- Representation of object surface characteristics (e.g. textures, materials)
- Taxonomies and aggregations
 - Digital Terrain Models as a combination of (including nested) triangulated irregular networks (TINs), regular rasters, break and skeleton lines, mass points
 - Sites (currently buildings, bridges, and tunnels)
 - Vegetation (areas, volumes, and solitary objects with vegetation classification)
 - Water bodies (volumes, surfaces)
 - Transportation facilities (both graph structures and 3D surface data)
 - Land use (representation of areas of the earth's surface dedicated to a specific land use)
 - City furniture
 - Generic city objects and attributes
 - User-definable (recursive) grouping
- Multiscale model with 5 well-defined consecutive Levels of Detail (LOD):
 - LOD0 – regional, landscape
 - LOD1 – city, region

- LOD2 – city districts, projects
- LOD3 – architectural models (outside), landmarks
- LOD4 – architectural models (interior)
- Multiple representations in different LODs simultaneously; generalisation relations between objects in different LODs
- Optional topological connections between feature (sub)geometries
- Application Domain Extensions (ADE): Specific “hooks” in the CityGML schema allow to define application specific extensions, for example for noise pollution simulation, or to augment CityGML by properties of the new National Building Information Model Standard (NBIMS) in the U.S.

Chapter 3. Conformance

This Best Practice defines XXXX.

Requirements for N target types are considered: * AAAA * BBBB

Conformance with this Best Practice shall be checked using all the relevant tests specified in Annex A (normative) of this document.

In order to conform to this OGC® Best Practice, a software implementation shall choose to implement: * Any one of the conformance levels specified in Annex A (normative). * Any one of the Distributed Computing Platform profiles specified in Annexes TBD through TBD (normative).

All requirements-classes and conformance-classes described in this document are owned by the document(s) identified.

NOTE the following is the 2.0 content

Conformance targets addressed by this International standard are CityGML instance documents only. Future revisions of this International Standard may also address consumers or producers as conformance targets. Clauses 8 to 10 of this International standard specify separate CityGML XML Schema definitions and normative aspects, i.e. CityGML modules, which shall be used in CityGML instance documents in accordance with clause 7. Implementations are not required to support the full range of capabilities provided by the universe of all CityGML modules. Valid partial implementations are supported following the rules and guidelines for CityGML profiles in chapter 7.2. CityGML instance documents claiming conformance to this International Standard shall: a) conform to the rules and requirements specified in clauses 7 to 10; b) pass all relevant test cases of the abstract test suite in annex B.1; c) satisfy all relevant conformance classes of the abstract test suite related to CityGML modules in annex B.2.□

Chapter 4. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of OGC 12-019. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 12-019 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

The following documents are indispensable for the application of the CityGML standard. The geometry model of GML 3.1.1 is used except for some added concepts like implicit geometries (cf. chapter 8.2). The appearance model (cf. chapter 9) draws concepts from both *X3D* and *COLLADA*. Addresses are represented using the OASIS extensible Address Language *xAL*.

- ISO / TC154: ISO 8601:2004, Data elements and interchange formats – Information interchange – Representation of dates and times
- ISO / TC211: ISO/TS 19103:2005, Geographic Information – Conceptual Schema Language
- ISO / TC211: ISO 19105:2000, Geographic information – Conformance and testing
- ISO / TC211: ISO 19107:2003, Geographic Information – Spatial Schema
- ISO / TC211: ISO 19109:2005, Geographic Information – Rules for Application Schemas
- ISO / TC211: ISO 19111:2003, Geographic information – Spatial referencing by coordinates
- ISO / TC211: ISO 19115:2003, Geographic Information – Metadata
- ISO / TC211: ISO 19123:2005, Geographic Information – Coverages
- ISO / TC211: ISO/TS 19139:2007, Geographic Information – Metadata – XML schema implementation
- ISO / TC211: ISO/IEC 19775:2004, X3D Abstract Specification
- OGC: OpenGIS® Abstract Specification Topic 0, Overview, OGC document 04-084
- OGC: OpenGIS® Abstract Specification Topic 5, The OpenGIS Feature, OGC document 99-105r2
- OGC: OpenGIS® Abstract Specification Topic 8, Relations between Features, OGC document 99-108r2
- OGC: OpenGIS® Abstract Specification Topic 10, Feature Collections, OGC document 99-110
- OGC: OpenGIS® Geography Markup Language Implementation Specification, Version 3.1.1, OGC document 03-105r1
- OGC: OpenGIS® GML 3.1.1 Simple Dictionary Profile, Version 1.0.0, OGC document 05-099r2
- IETF: IETF RFC 2045 & 2046, Multipurpose Internet Mail Extensions (MIME). (November 1996)
- IETF: IETF RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax. (August 1998)
- W3C: W3C XLink, XML Linking Language (XLink) Version 1.0. W3C Recommendation (27 June 2001)
- W3C: W3C XMLName, Namespaces in XML. W3C Recommendation (14 January 1999)
- W3C: W3C XMLSchema-1, XML Schema Part 1: Structures. W3C Recommendation (2 May 2001)

- W3C: W3C XMLSchema-2, XML Schema Part 2: Datatypes. W3C Recommendation (2 May 2001)
- W3C: W3C XPointer, XML Pointer Language (XPointer) Version 1.0. W3C Working Draft (16 August 2002)
- W3C: W3C XML Base, XML Base, W3C Recommendation (27 June 2001)
- W3C: W3C XML, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)
- OASIS (Organization for the Advancement of Structured Information Standards): extensible Address Language (xAL v2.0).
- Khronos Group Inc.: COLLADA – Digital Asset Schema Release 1.4.1
- Jelliffe, R: The Schematron Assertion Language 1.5. (2002-10-01)

Chapter 5. 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.

5.1. term name

text of the definition

5.2. Abbreviated Terms

The following abbreviated terms are used in this document:

- 2D Two Dimensional
- 3D Three Dimensional
- AEC Architecture, Engineering, Construction
- ALKIS German National Standard for Cadastral Information
- ATKIS German National Standard for Topographic and Cartographic Information
- B-Rep Boundary Representation
- bSI buildingSMART International
- CAD Computer Aided Design
- COLLADA Collaborative Design Activity
- CSG Constructive Solid Geometry
- DTM Digital Terrain Model
- DXF Drawing Exchange Format
- EuroSDR European Spatial Data Research Organisation
- ESRI Environmental Systems Research Institute
- FM Facility Management
- GDF Geographic Data Files
- GDI-DE Spatial Data Infrastructure Germany (Geodateninfrastruktur Deutschland)
- GDI NRW Geodata Infrastructure North-Rhine Westphalia
- GML Geography Markup Language
- IAI International Alliance for Interoperability (now buildingSMART International (bSI))
- IETF Internet Engineering Task Force
- IFC Industry Foundation Classes

- ISO International Organization for Standardisation
- LOD Level of Detail
- NBIMS National Building Information Model Standard
- OASIS Organisation for the Advancement of Structured Information Standards
- OGC Open Geospatial Consortium
- OSCRE Open Standards Consortium for Real Estate
- SIG 3D Special Interest Group 3D of the GDI-DE
- TC211 ISO Technical Committee 211
- TIC Terrain Intersection Curve
- TIN Triangulated Irregular Network
- UML Unified Modeling Language
- URI Uniform Resource Identifier
- VRML Virtual Reality Modeling Language
- W3C World Wide Web Consortium
- W3DS OGC Web 3D Service
- WFS OGC Web Feature Service
- X3D Open Standards XML-enabled 3D file format of the Web 3D Consortium
- XML Extensible Markup Language
- xAL OASIS extensible Address Language

Chapter 6. Conventions

6.1. Identifiers

The normative provisions in this document are denoted by the URI

<http://www.opengis.net/spec/CityGML/3.0>

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

6.2. UML Notation

The CityGML standard is presented in this document in diagrams using the Unified Modeling Language (UML) static structure diagram (see Booch et al. 1997). The UML notations used in this standard are described in the diagram below [Figure 1](#).

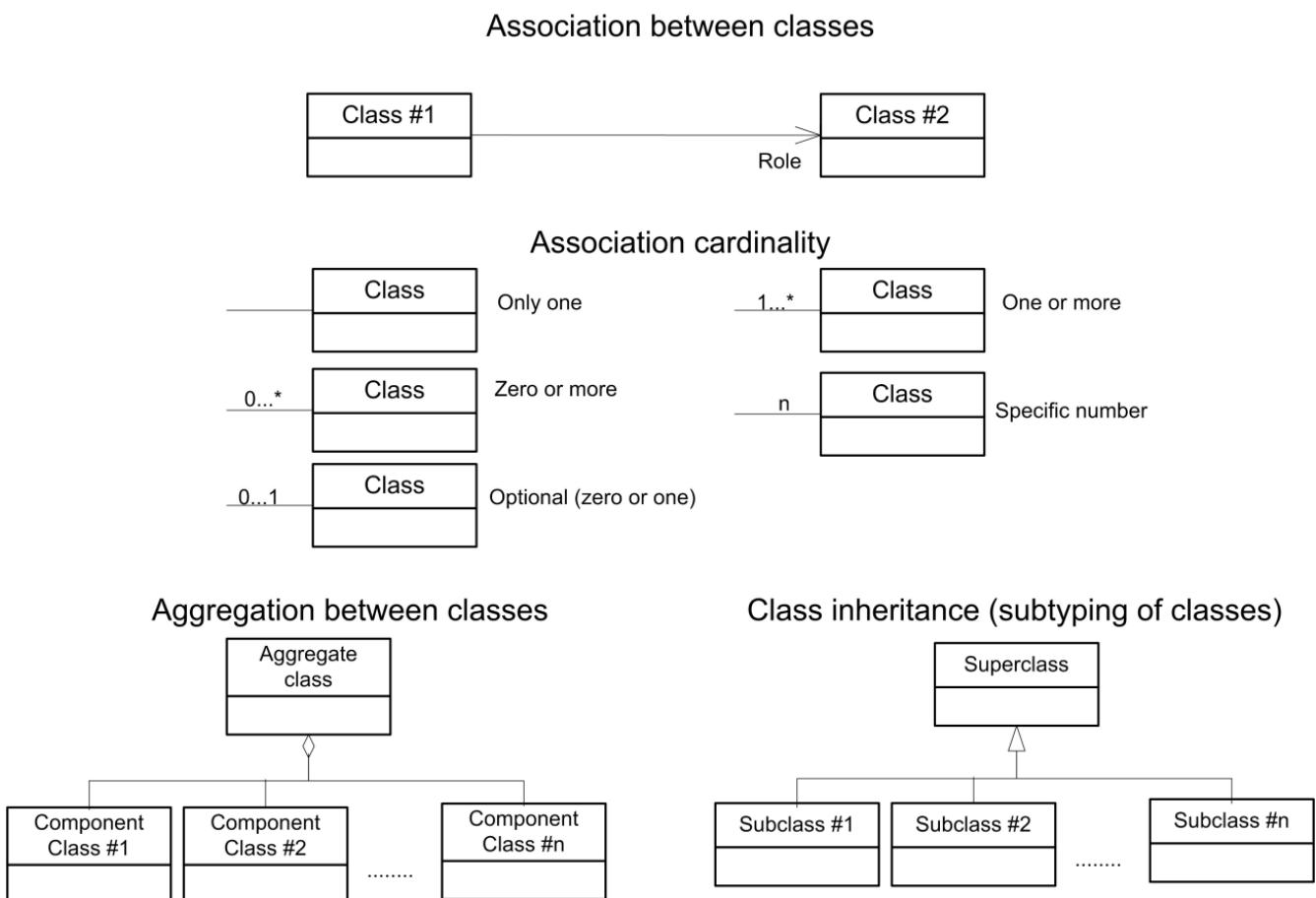


Figure 1. UML notation (see ISO TS 19103, Geographic information - Conceptual schema language).

According to GML3 all associations between model elements in CityGML are uni-directional. Thus, associations in CityGML are navigable in only one direction. The direction of navigation is depicted by an arrowhead. In general, the context an element takes within the association is indicated by its role. The role is displayed near the target of the association. If the graphical representation is ambiguous though, the position of the role has to be drawn to the element the association points to.

The following stereotypes are used:

- <<Geometry>> represents the geometry of an object. The geometry is an identifiable and distinguishable object that is derived from the abstract GML type *AbstractGeometryType*.
- <<Feature>> represents a thematic feature according to the definition in ISO 19109. A feature is an identifiable and distinguishable object that is derived from the abstract GML type *AbstractFeatureType*.
- <<Object>> represents an identifiable and distinguishable object that is derived from the abstract GML type *AbstractGMLType*.
- <<Enumeration>> enumerates the valid attribute values in a fixed list of named literal values. Enumerations are specified inline the CityGML schema.
- <<CodeList>> enumerates the valid attribute values. In contrast to Enumeration, the list of values is open and, thus, not given inline the CityGML schema. The allowed values can be provided within an external code list. It is recommended that code lists are implemented as simple dictionaries following the GML 3.1.1 Simple Dictionary Profile (cf. chapter 6.6 and chapter 10.14).
- <<Union>> is a list of attributes. The semantics are that only one of the attributes can be present at any time.
- <<PrimitiveType>> is used for representations supported by a primitive type in the implementation.
- <<DataType>> is used as a descriptor of a set of values that lack identity. Data types include primitive pre-defined types and user-definable types. A DataType is thus a class with few or no operations whose primary purpose is to hold the abstract state of another class for transmittal, storage, encoding or persistent storage.
- <<Leaf>> is used within UML package diagrams to indicate model elements that can have no further subtypes.
- <<XSDSchema>> is used within UML package diagrams to denote the root element of an XSD Schema containing all the definitions for a particular namespace. All the package contents or component classes are placed within the one schema.
- <<ApplicationSchema>> is used within UML package diagrams to denote an XML Schema definition fundamentally dependent on the concepts of another independent Standard within the XML Schema metalinguage. For example, ApplicationSchema indicates extensions of GML consistent with the GML “rules for application schemas”.

In order to enhance the readability of the CityGML UML diagrams, classes are depicted in different colors if they belong to different UML packages (see Fig. 8 for an overview of UML packages). The following coloring scheme is applied:

- Classes painted in yellow belong to the UML package which is subject of discussion in that clause of the specification in which the UML diagram is given. For example, in the context of chapter 10.1 which introduces the *CityGML Core* module, the yellow color is used to denote classes which are defined in the *CityGML Core* UML package. Likewise, the yellow classes shown in UML diagrams in chapter 10.3 are associated with the *Building* module which is subject of discussion in that chapter.

- Classes painted in blue belong to a CityGML UML package different to that associated with the yellow color. In order to explicitly denote the UML package of such classes, their class names carry a namespace prefix which is uniquely associated with a CityGML module throughout this specification (cf. section 4.3 for a list of namespaces and prefixes). For example, in the context of the *Building* module, classes from the *CityGML Core* module are painted in blue and their class names are preceded by the prefix *core*.
- Classes painted in green are defined in GML3 and their class names are preceded by the prefix *gml*.

The following example UML diagram demonstrates the UML notation and coloring scheme used throughout this specification. In this example, the yellow classes are associated with the *CityGML Building* module, the blue classes are from the *CityGML Core* module, and the green class depicts a geometry element defined by GML3.

Visual Paradigm for UML Standard Edition (Technische Universität Berlin)

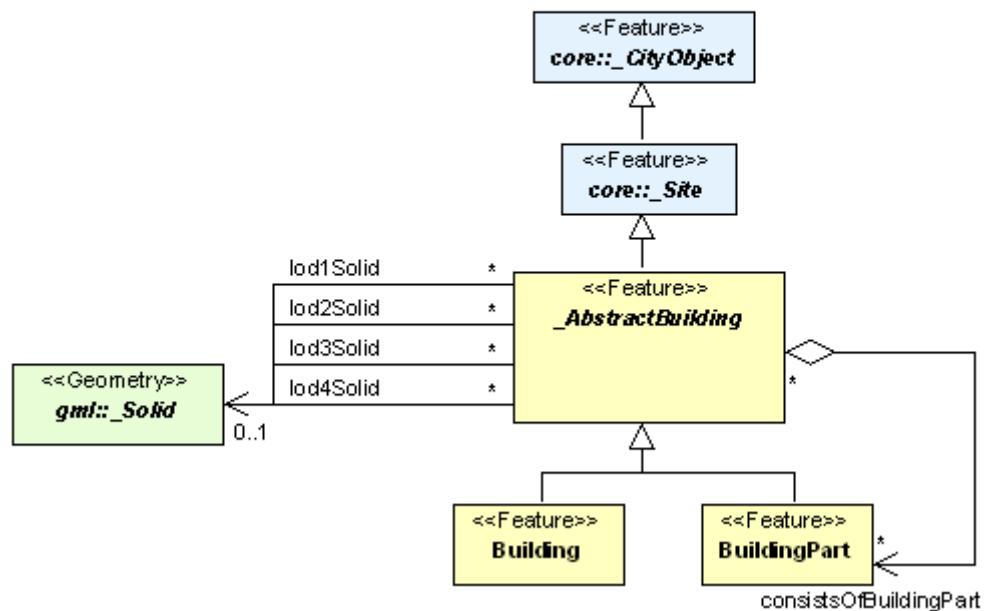


Figure 2. Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML specification.

6.3. XML namespaces and namespace prefixes

The CityGML data model is thematically decomposed into a core module and thematic extension modules. All modules including the core are specified by their own XML schema file, each defining a globally unique XML namespace. The extension modules are based on the core module and, thus, contain (by reference) the CityGML core schema.

Within this document the module namespaces are associated with recommended prefixes. These prefixes are consistently used within the normative parts of this specification, for all UML diagrams and example CityGML instance documents. The CityGML core and extension modules along with their XML namespace identifiers and recommended namespace prefixes are listed in Tab. 1.

Table 3. List of CityGML modules, their associated XML namespace identifiers, and example namespace prefixes.

CityGML module	Namespace identifier	Namespace prefix
CityGML Core	http://www.opengis.net/citygml/2.0	core
Appearance	http://www.opengis.net/citygml/appearance/2.0	app
Bridge	http://www.opengis.net/citygml/bridge/2.0	brid
Building	http://www.opengis.net/citygml/building/2.0	bldg
CityFurniture	http://www.opengis.net/citygml/cityfurniture/2.0	frn
CityObjectGroup	http://www.opengis.net/citygml/cityobjectgroup/2.0	grp
Generics	http://www.opengis.net/citygml/generics/2.0	gen
LandUse	http://www.opengis.net/citygml/landuse/2.0	luse
Relief	http://www.opengis.net/citygml/relief/2.0	dem
Transportation	http://www.opengis.net/citygml/transportation/2.0	tran
Tunnel	http://www.opengis.net/citygml/tunnel/2.0	tun
Vegetation	http://www.opengis.net/citygml/vegetation/2.0	veg
WaterBody	http://www.opengis.net/citygml/waterbody/2.0	wtr
TexturedSurface [deprecated]	http://www.opengis.net/citygml/texturedsurface/2.0	tex

Further XML Schema definitions relevant to this standard are shown in Tab. 2 along with the corresponding XML namespace identifiers and namespace prefixes consistently used within this document.

Table 4. List of XML Schema definitions, their associated XML namespace identifiers, and example namespace prefixes used within this document.

XML Schema Definition	Namespace identifier	Namespace prefix
Geography Markup Language version 3.1.1 (from OGC)	http://www.opengis.net/gml	gml

XML Schema Definition	Namespace identifier	Namespace prefix
Extensible Address Language version 2.0 (from OASIS)	urn:oasis:names:tc:ciq:xsdschema: xAL:2.0	xAL
Schematron Assertion Lan-guage version 1.5	http://www.ascc.net/xml/ schematron	sch

Chapter 7. Overview of CityGML

CityGML is an open data model and XML-based format for the storage and exchange of virtual 3D city models. It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211.

The aim of the development of CityGML is to reach a common definition of the basic entities, attributes, and relations of a 3D city model. This is especially important with respect to the cost-effective sustainable maintenance of 3D city models, allowing the reuse of the same data in different application fields.

CityGML not only represents the graphical appearance of city models but specifically addresses the representation of the semantic and thematic properties, taxonomies and aggregations. CityGML includes a geometry model and a thematic model. The geometry model allows for the consistent and homogeneous definition of geometrical and topological properties of spatial objects within 3D city models (chapter 8). The base class of all objects is *_CityObject* which is a subclass of the GML class *_Feature*. All objects inherit the properties from *_CityObject*.

The thematic model of CityGML employs the geometry model for different thematic fields like Digital Terrain Models, sites (i.e. buildings, bridges, and tunnels), vegetation (solitary objects and also areal and volumetric biotopes), land use, water bodies, transportation facilities, and city furniture (chapter 10). Further objects, which are not explicitly modelled yet, can be represented using the concept of generic objects and attributes (chapter 6.11). In addition, extensions to the CityGML data model applying to specific application fields can be realised using the Application Domain Extensions (ADE) (chapter 6.12). Spatial objects of equal shape which appear many times at different positions like e.g. trees, can also be modelled as prototypes and used multiple times in the city model (chapter 8.2). A grouping concept allows the combination of single 3D objects, e.g. buildings to a building complex (chapter 6.8). Objects which are not geometrically modelled by closed solids can be virtually sealed in order to compute their volume (e.g. pedestrian underpasses, tunnels, or airplane hangars). They can be closed using *ClosureSurfaces* (chapter 6.4). The concept of the *TerrainIntersectionCurve* is introduced to integrate 3D objects with the Digital Terrain Model at their correct positions in order to prevent e.g. buildings from floating over or sinking into the terrain (chapter 6.5).

CityGML differentiates five consecutive Levels of Detail (LOD), where objects become more detailed with increasing LOD regarding both their geometry and thematic differentiation (chapter 6.2). CityGML files can - but do not have to - contain multiple representations (and geometries) for each object in different LOD simultaneously. Generalisation relations allow the explicit representation of aggregated objects over different scales.

In addition to spatial properties, CityGML features can be assigned appearances. Appearances are not limited to visual data but represent arbitrary observable properties of the feature's surface such as infrared radiation, noise pollution, or earthquake-induced structural stress (chapter 9).

Furthermore, objects can have external references to corresponding objects in external datasets (chapter 6.7). The possible attribute values of enumerative object attributes can be enumerated in code lists defined in external, redefinable dictionaries (chapter 6.6).

Chapter 8. General characteristics of CityGML

8.1. Modularisation

The CityGML data model consists of class definitions for the most important types of objects within virtual 3D city models. These classes have been identified to be either required or important in many different application areas. However, implementations are not required to support the overall CityGML data model in order to be conformant to the standard, but may employ a subset of constructs according to their specific information needs. For this purpose, modularisation is applied to the CityGML data model (cf. chapter 7).

The CityGML data model is thematically decomposed into a *core module* and thematic *extension modules*. The core module comprises the basic concepts and components of the CityGML data model and, thus, must be implemented by any conformant system. Based on the core module, each extension covers a specific thematic field of virtual 3D city models. CityGML introduces the following thirteen thematic extension modules: *Appearance*, *Bridge*, *Building*, *CityFurniture*, *CityObjectGroup*, *Generics*, *LandUse*, *Relief*, *Transportation*, *Tunnel*, *Vegetation*, *WaterBody*, and *TexturedSurface* [deprecated].

CityGML compliant implementations may support any combination of extension modules in conjunction with the core module. Such combinations of modules are called CityGML profiles. Therefore, CityGML profiles allow for valid partial implementations of the overall CityGML data model.

8.2. Multi-scale modelling (5 levels of detail, LOD)

CityGML supports different Levels of Detail (LOD). LODs are required to reflect independent data collection processes with differing application requirements. Further, LODs facilitate efficient visualisation and data analysis (see Fig. 3). In a CityGML dataset, the same object may be represented in different LOD simultaneously, enabling the analysis and visualisation of the same object with regard to different degrees of resolution. Furthermore, two CityGML data sets containing the same object in different LOD may be combined and integrated. However, it will be within the responsibility of the user or application to make sure objects in different LODs refer to the same real-world object.

The coarsest level LOD0 is essentially a two and a half dimensional Digital Terrain Model over which an aerial image or a map may be draped. Buildings may be represented in LOD0 by footprint or roof edge polygons. LOD1 is the well-known blocks model comprising prismatic buildings with flat roof structures. In contrast, a building in LOD2 has differentiated roof structures and thematically differentiated boundary surfaces. LOD3 denotes architectural models with detailed wall and roof structures potentially including doors and windows. LOD4 completes a LOD3 model by adding interior structures for buildings. For example, buildings in LOD4 are composed of rooms, interior doors, stairs, and furniture. In all LODs appearance information such as highresolution textures can be mapped onto the structures (cf. 6.9).

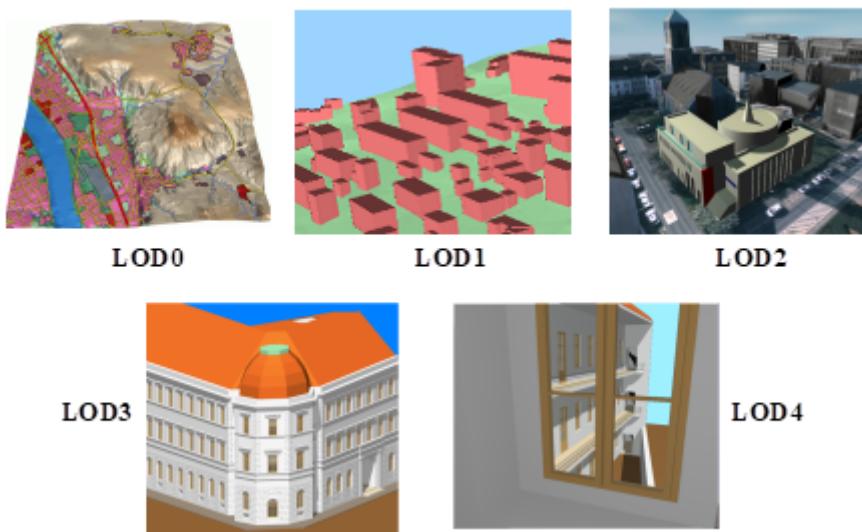


Figure 3. The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn)

LODs are also characterised by differing accuracies and minimal dimensions of objects (cf. Tab. 3). The accuracy requirements given in this standard are debatable and are to be considered as discussion proposals. Accuracy is described as standard deviation σ of the absolute 3D point coordinates. Relative 3D point accuracy will be added in a future version of CityGML and it is typically much higher than the absolute accuracy. In LOD1, the positional and height accuracy of points should be 5m or less, while all objects with a footprint of at least 6m by 6m should be considered. The positional and height accuracy of LOD2 is proposed to be 2m or better. In this LOD, all objects with a footprint of at least 4m \times 4m should be considered. Both types of accuracies in LOD3 should be 0.5m, and the minimal footprint is suggested to be 2m \times 2m. Finally, the positional and height accuracy of LOD4 should be 0.2m or less. By means of these figures, the classification in five LOD may be used to assess the quality of 3D city model datasets. The LOD categorisation makes datasets comparable and provides support for their integration.

Table 5. LOD 0-4 of CityGML with their proposed accuracy requirements (discussion proposal, based on: Albert et al. 2003).

	LOD0	LOD1	LOD2	LOD3	LOD4
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	very high
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m

	LOD0	LOD1	LOD2	LOD3	LOD4
Generalisation	maximal generalisation	object blocks as generalised features; > 6*6m/3m	objects as generalised features; > 4*4m/2m	object as real features; > 2*2m/1m	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure/representation	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
CityFurniture	no	important objects	prototypes, generalized objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	no	>50*50m	>5*5m	< LOD2	<LOD2
...to be continued for the other feature themes					

Whereas in CityGML each object can have a different representation for every LOD, often different objects from the same LOD will be generalised to be represented by an aggregate object in a lower LOD. CityGML supports the aggregation / decomposition by providing an explicit generalisation association between city objects (further details see UML diagram in chapter 10.1).

8.3. Coherent semantical-geometrical modelling

One of the most important design principles for CityGML is the coherent modelling of semantics and geometrical/topological properties. At the semantic level, real-world entities are represented by features, such as buildings, walls, windows, or rooms. The description also includes attributes, relations and aggregation hierarchies (part-whole-relations) between features. Thus the part-of-relationship between features can be derived at the semantic level only, without considering geometry. However, at the spatial level, geometry objects are assigned to features representing their spatial location and extent. So the model consists of two hierarchies: the semantic and the geometrical in which the corresponding objects are linked by relationships (cf. Stadler & Kolbe 2007). The advantage of this approach is that it can be navigated in both hierarchies and between both hierarchies arbitrarily, for answering thematic and/or geometrical queries or performing analyses.

If both hierarchies exist for a specific object, they must be coherent (i.e. it must be ensured that they match and fit together). For example, if a wall of a building has two windows and a door on the semantic level, then the geometry representing the wall must contain also the geometry parts of both windows and the door.

8.4. Closure surfaces

Objects, which are not modelled by a volumetric geometry, must be virtually closed in order to compute their volume (e.g. pedestrian underpasses or airplane hangars). They can be sealed using a ClosureSurface. These are special surfaces, which are taken into account, when needed to compute volumes and are neglected, when they are irrelevant or not appropriate, for example in visualisations.

The concept of ClosureSurface is also employed to model the entrances of subsurface objects. Those objects like tunnels or pedestrian underpasses have to be modelled as closed solids in order to compute their volume, for example in flood simulations. The entrances to subsurface objects also have to be sealed to avoid holes in the digital terrain model [Figure 4](#). However, in close-range visualisations the entrance must be treated as open. Thus, closure surfaces are an adequate way to model those entrances.

NOTE Combine Figures 4



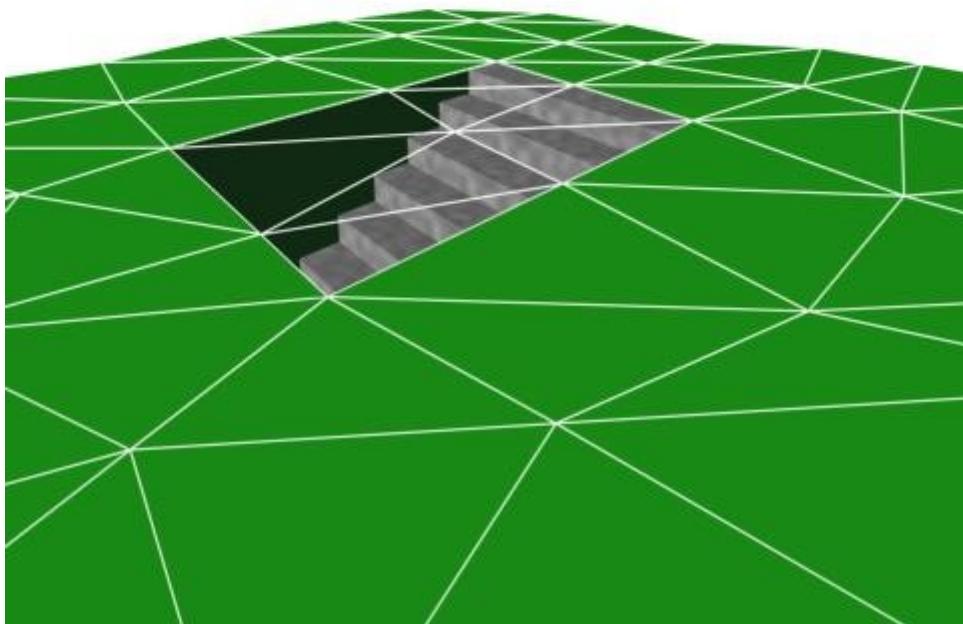


Figure 4. Closure surfaces to seal open structures. Passages are subsurface objects (left). The entrance is sealed by a virtual ClosureSurface, which is both part of the DTM and the subsurface object (right) (graphic: IGG Uni Bonn).

8.5. Terrain Intersection Curve (TIC)

A crucial issue in city modelling is the integration of 3D objects and the terrain. Problems arise if 3D objects float over or sink into the terrain. This is particularly the case if terrains and 3D objects in different LOD are combined, or if they come from different providers (Kolbe and Gröger 2003). To overcome this problem, the TerrainIntersectionCurve (TIC) of a 3D object is introduced. These curves denote the exact position, where the terrain touches the 3D object (see Fig. 5). TICs can be applied to buildings and building parts (cf. chapter 10.3), bridge, bridge parts and bridge construction elements (cf. chapter 10.5), tunnel and tunnel parts (cf. chapter 10.4), city furniture objects (cf. chapter 10.9), and generic city objects (cf. chapter 10.12). If, for example, a building has a courtyard, the TIC consists of two closed rings: one ring representing the courtyard boundary, and one which describes the building's outer boundary. This information can be used to integrate the building and a terrain by ‘pulling up’ or ‘pulling down’ the surrounding terrain to fit the TerrainIntersectionCurve. The DTM may be locally warped to fit the TIC. By this means, the TIC also ensures the correct positioning of textures or the matching of object textures with the DTM. Since

the intersection with the terrain may differ depending on the LOD, a 3D object may have different TerrainIntersectionCurves for all LOD.

NOTE Combine figures 5





Figure 5. *TerrainIntersectionCurve* for a building (left, black) and a tunnel object (right, white). The tunnel's hollow space is sealed by a triangulated *ClosureSurface* (graphic: IGG Uni Bonn).

8.6. Code lists for enumerative attributes

CityGML feature types often include attributes whose values can be enumerated in a list of discrete values. An example is the attribute roof type of a building, whose attribute values typically are saddle back roof, hip roof, semi-hip roof, flat roof, pent roof, or tent roof. If such an attribute is typed as string, misspellings or different names for the same notion obstruct interoperability. Moreover, the list of possible attribute values often is not fixed and may substantially vary for different countries (e.g., due to national law and regulations) and for different information communities.

In CityGML, such enumerative attributes are of type `gml:CodeType` and their allowed attribute values can be provided in a code list which is specified outside the CityGML schema. A code list contains coded attribute values and ensures that the same code is used for the same notion or concept. If a code list is provided for an enumerative attribute, the attribute may only take values from this list. This allows applications to validate the attribute value and thus facilitates semantic and syntactic interoperability. It is recommended that code lists are implemented as simple dictionaries following the GML 3.1.1 Simple Dictionary Profile (cf. Whiteside 2005).

The governance of code lists is decoupled from the governance of the CityGML schema and specification. Thus, code lists may be specified by any organisation or information community according to their information needs. There shall be one authority per code list who is in charge of the code list values and the maintenance of the code list. Further information on the CityGML code

list mechanism is provided in chapter 10.14.

Code lists can have references to existing models. For example, room codes defined by the Open Standards Consortium for Real Estate (OSCRE) can be referenced or classifications of buildings and building parts introduced by the National Building Information Model Standard (NBIMS) can be used. Annex C contains non-normative code lists proposed by the SIG 3D for almost all enumerative attributes in CityGML. They can be directly referenced in CityGML instance documents and serve as an example for the definition of code lists.

8.7. External references

3D objects are often derived from or have relations to objects in other databases or data sets. For example, a 3D building model may have been constructed from a two-dimensional footprint in a cadastre data set, or may be derived from an architectural model (Fig. 6). The reference of a 3D object to its corresponding object in an external data set is essential, if an update must be propagated or if additional data is required, for example the name and address of a building's owner in a cadastral information system or information on antennas and doors in a facility management system. In order to supply such information, each `_CityObject` may refer to external data sets (for the UML diagram see Fig. 21; and for XML schema definition see annex A.1) using the concept of `ExternalReference`. Such a reference denotes the external information system and the unique identifier of the object in this system. Both are specified as a Uniform Resource Identifier (URI), which is a generic format for references to any kind of resources on the internet. The generic concept of external references allows for any `_CityObject` an arbitrary number of links to corresponding objects in external information systems (e.g. ALKIS, ATKIS, OS MasterMap®, GDF, etc.).

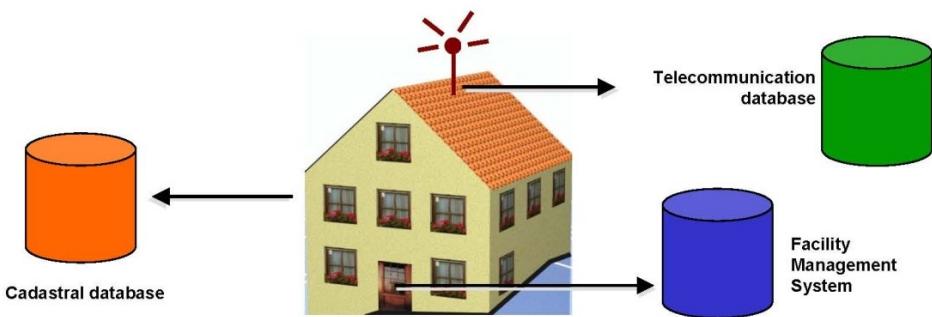


Figure 6. External references (graphic: IGG Uni Bonn).

8.8. City object groups

The grouping concept of CityGML allows for the aggregation of arbitrary city objects according to user-defined criteria, and to represent and transfer these aggregations as part of a city model (for the UML diagram see chapter 10.11; XML schema definition see annex A.6). A group may be assigned one or more names and may be further classified by specific attributes, for example, "escape route from room no. 43 in house no. 1212 in a fire scenario" as a name and "escape route" as type. Each member of the group can optionally be assigned a role name, which specifies the role this particular member plays in the group. This role name may, for example, describe the sequence number of this object in an escape route, or in the case of a building complex, denote the main building.

A group may contain other groups as members, allowing nested grouping of arbitrary depth. The grouping concept is delivered by the thematic extension module `CityObjectGroup` of CityGML (cf. chapter 10.11).

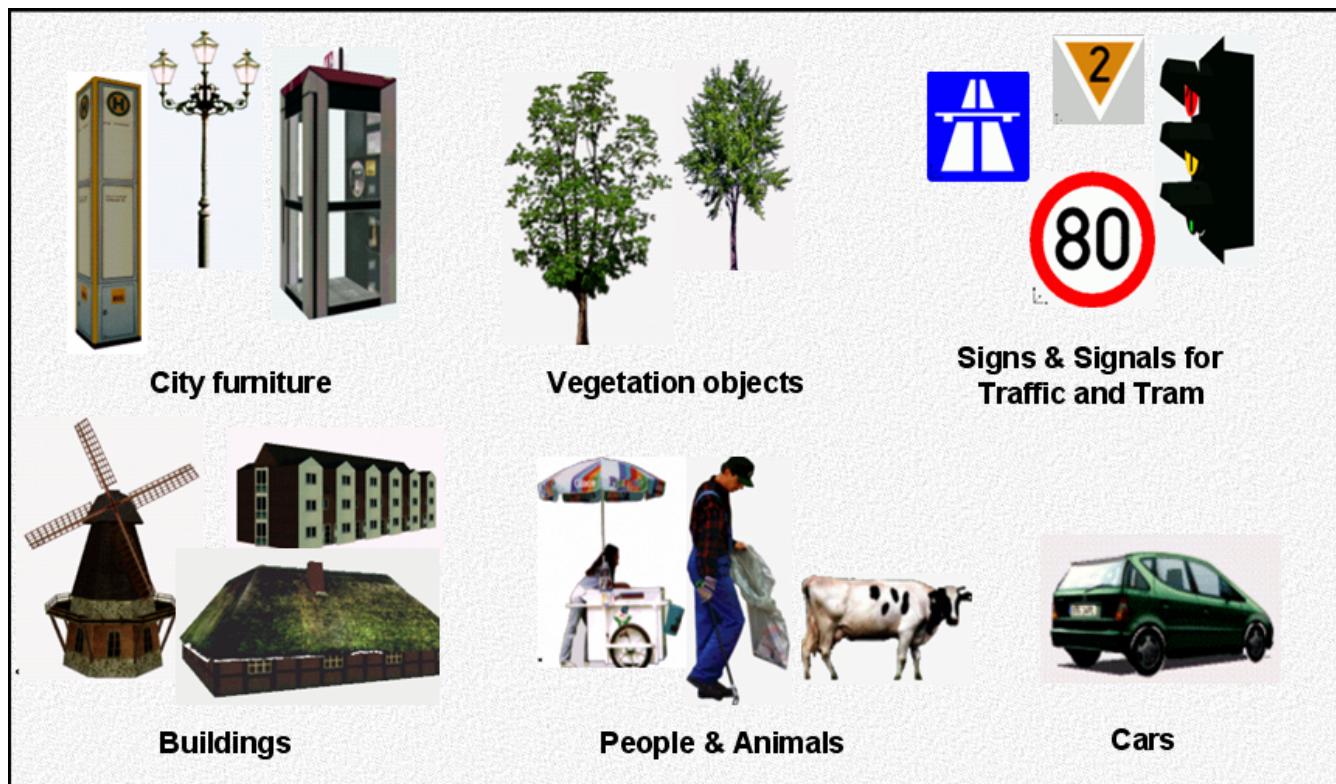
8.9. Appearances

Information about a surface's appearance, i.e. observable properties of the surface, is considered an integral part of virtual 3D city models in addition to semantics and geometry. Appearance relates to any surface-based theme, e.g. infrared radiation or noise pollution, not just visual properties. Consequently, data provided by appearances can be used as input for both presentation of and analysis in virtual 3D city models.

CityGML supports feature appearances for an arbitrary number of themes per city model. Each LOD of a feature can have an individual appearance. Appearances can represent – among others – textures and georeferenced textures. CityGML’s appearance model is packaged within its own extension module Appearance (cf. chapter 9).

8.10. Prototypic objects / scene graph concepts

In CityGML, objects of equal shape like trees and other vegetation objects, traffic lights and traffic signs can be represented as prototypes which are instantiated multiple times at different locations (Fig. 7). The geometry of prototypes is defined in local coordinate systems. Every instance is represented by a reference to the prototype, a base point in the world coordinate reference system and a transformation matrix that facilitates scaling, rotation, and translation of the prototype. The principle is adopted from the concept of scene graphs used in computer graphics standards like VRML and X3D. As the GML3 geometry model does not provide support for scene graph concepts, it is implemented as an extension to the GML3 geometry model (for further description cf. chapter 8.2).



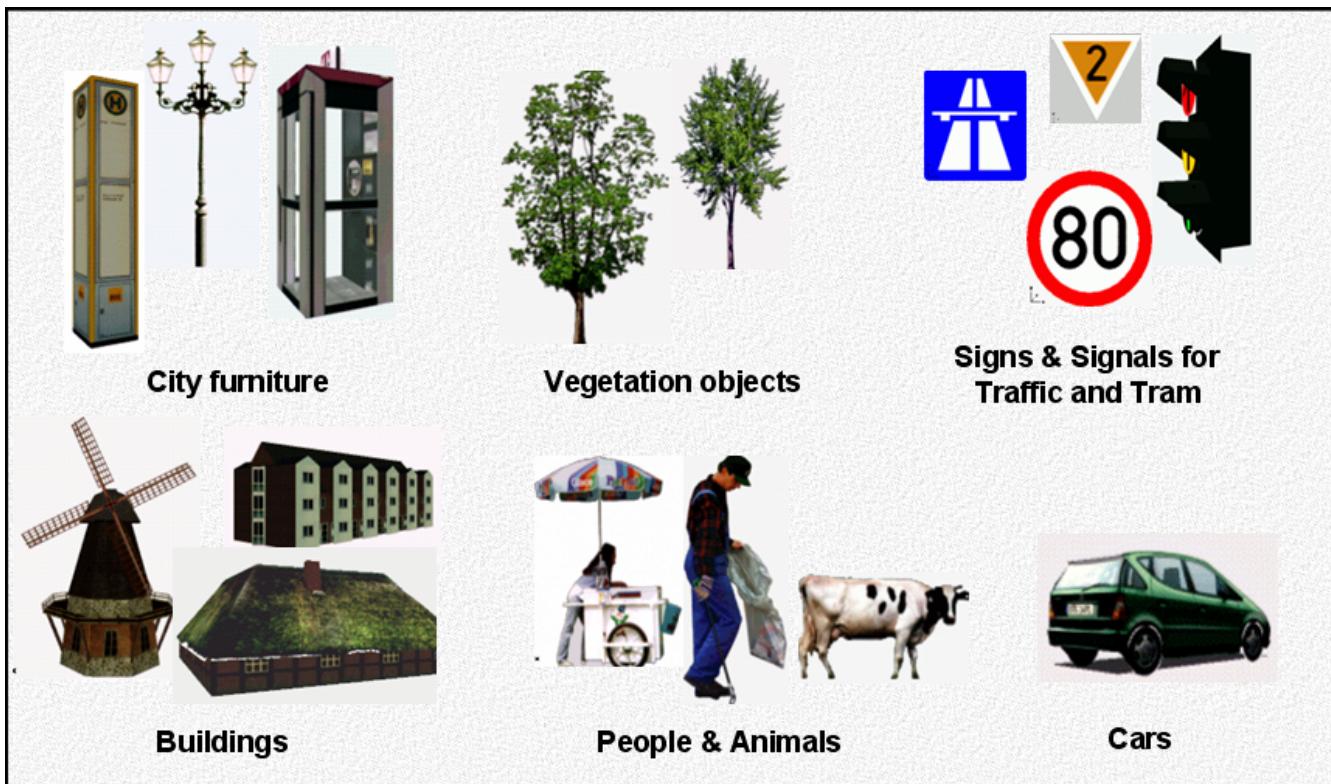


Figure 7. Examples of prototypic shapes (source: Rheinmetall Defence Electronics).

8.11. Generic city objects and attributes

CityGML is being designed as a universal topographic information model that defines object types and attributes which are useful for a broad range of applications. In practical applications the objects within specific 3D city models will most likely contain attributes which are not explicitly modelled in CityGML. Moreover, there might be 3D objects which are not covered by the thematic classes of CityGML. CityGML provides two different concepts to support the exchange of such data: 1) generic objects and attributes, and 2) Application Domain Extensions (cf. chapter 6.12).

The concept of generic objects and attributes allows for the extension of CityGML applications during runtime, i.e. any _CityObject may be augmented by additional attributes, whose names, data types, and values can be provided by a running application without any change of the CityGML XML schema. Similarly, features not represented by the predefined thematic classes of the CityGML data model may be modelled and exchanged using generic objects. The generic extensions of CityGML are provided by the thematic extension module Generics (cf. chapter 10.12).

The current version of CityGML does not include, for example, explicit thematic models for embankments, excavations and city walls. These objects may be stored or exchanged using generic objects and attributes.

8.12. Application Domain Extensions (ADE)

Application Domain Extensions (ADE) specify additions to the CityGML data model. Such additions comprise the introduction of new properties to existing CityGML classes like e.g. the number of habitants of a building or the definition of new object types. The difference between ADEs and generic objects and attributes is, that an ADE has to be defined in an extra XML schema definition file with its own namespace. This file has to explicitly import the XML Schema definition of the

extended CityGML modules.

The advantage of this approach is that the extension is formally specified. Extended CityGML instance documents can be validated against the CityGML and the respective ADE schema. ADEs can be defined (and even standardised) by information communities which are interested in specific application fields. More than one ADE can be actively used in the same dataset (further description cf. chapter 10.13).

ADEs may be defined for one or even several CityGML modules providing a high flexibility in adding additional information to the CityGML data model. Thus, the ADE mechanism is orthogonally aligned with the modularisation approach of CityGML. Consequently, there is no separate extension module for ADEs.

In this specification, two examples for ADEs are included:

- An ADE for Noise Immission Simulation (Annex H) which is employed in the simulation of environmental noise dispersion according to the Environmental Noise Directive of the European Commission (2002/49/EC);
- An ADE for Ubiquitous Network Robots Services (Annex I) which demonstrates the usage of CityGML for the navigation of robots in indoor environments.

Further examples for ADEs are the CAFM ADE (Bleifuß et al., 2009) for facility management, the UtilityNetworkADE (Becker et al., 2011) for the integrated 3D modeling of multi-utility networks and their interdependencies, the HydroADE (Schulte and Coors, 2008) for hydrographical applications and the GeoBIM (IFC) ADE (van Berlo et al., 2011) which combines BIM information from IFC (from bSI) with CityGML and is implemented in the open source modelserver BIMserver.org.

Chapter 9. Modularization

CityGML is a rich standard both on the thematic and geometric-topological level of its data model. On its thematic level CityGML defines classes and relations for the most relevant topographic objects in cities and regional models comprising built structures, elevation, vegetation, water bodies, city furniture, and more. In addition to geometry and appearance content these thematic components allow to employ virtual 3D city models for sophisticated analysis tasks in different application domains like simulations, urban data mining, facility management, and thematic inquiries.

CityGML is to be seen as a framework giving geospatial 3D data enough space to grow in geometrical, topographical and semantic aspects over its lifetime. Thus, geometry and semantics of city objects may be flexibly structured covering purely geometric datasets up to complex geometric-topologically sound and spatio-semantically coherent data. By this means, CityGML defines a single object model and data exchange format applicable to consecutive process steps of 3D city modelling from geometry acquisition, data qualification and refinement to preparation of data for specific end-user applications, allowing for iterative data enrichment and lossless information exchange (cf. Kolbe et al. 2009).

According to this idea of a framework, applications are not required to support all thematic fields of CityGML in order to be compliant to the standard, but may employ a subset of constructs corresponding to specific relevant requirements of an application domain or process step. The use of logical subsets of CityGML limits the complexity of the overall data model and explicitly allows for valid partial implementations. As for version 2.0 of the CityGML standard, possible subsets of the data model are defined and embraced by so called CityGML modules. A CityGML module is an aggregate of normative aspects that must all be implemented as a whole by a conformant system. CityGML consists of a core module and thematic extension modules.

The CityGML core module defines the basic concepts and components of the CityGML data model. It is to be seen as the universal lower bound of the overall CityGML data model and a dependency of all thematic extension modules. Thus, the core module is unique and must be implemented by any conformant system. Based on the CityGML core module, each extension module contains a logically separate thematic component of the CityGML data model. The extensions to the core are derived by vertically slicing the overall CityGML data model. Since the core module is contained (by reference) in each extension module, its general concepts and components are universal to all extension modules. The following thirteen thematic extension modules are introduced by version 2.0 of the CityGML standard. They are directly related to clauses of this document each covering the corresponding thematic field of CityGML:

- Appearance (cf. clause 9),
- Bridge (cf. clause 10.5)
- Building (cf. clause 10.3),
- CityFurniture (cf. clause 10.9),
- CityObjectGroup (cf. clause 10.11),
- Generics (cf. clause 10.12),
- LandUse (cf. clause 10.10),

- Relief (cf. clause 10.2),
- Transportation (cf. clause 10.7),
- Tunnel (cf. clause 10.4)
- Vegetation (cf. clause 10.8),
- WaterBody (cf. clause 10.6), and
- TexturedSurface [deprecated] (cf. clause 9.8).

The thematic decomposition of the CityGML data model allows for implementations to support any combination of extension modules in conjunction with the core module in order to be CityGML conformant. Thus, the extension modules may be arbitrarily combined according to the information needs of an application or application domain. A combination of modules is called a CityGML profile. The union of all modules is defined as the CityGML base profile. The base profile is unique at any given time and forms the upper bound of the overall CityGML data model. Any other CityGML profile must be a valid subset of the base profile. By following the concept of CityGML modules and profiles, valid partial implementations of the CityGML data model may be realised in a well-defined way.

As for future development, each CityGML module may be further developed independently from other modules by expert groups and information communities. Resulting proposals and changes to modules may be introduced into future revisions of the CityGML standard without affecting the validity of other modules. Furthermore, thematic components not covered by the current CityGML data model may be added to future revisions of the standard by additional thematic extension modules. These additional extensions may establish dependency relations to any other existing CityGML module but shall at least be dependent on the CityGML core module. Consequently, the CityGML base profile may vary over time as new extensions are added. However, if a specific application has information needs to be modelled and exchanged which are beyond the scope of the CityGML data model, this application data can also be incorporated within the existing modules using CityGML's Application Domain Extension mechanism (cf. clause 10.13) or by employing the concepts of generic city objects and attributes (cf. chapter 10.12).

The introduced modularisation approach supports CityGML's versatility as a data modelling framework and exchange format addressing various application domains and different steps of 3D city modelling. For sake of clarity, applications should announce the level of conformance to the CityGML standard by declaring the employed CityGML profile. Since the core module is part of all profiles, this should be realised by enumerating the implemented thematic extension modules. For example, if an implementation supports the Building module, the Relief module, and the Vegetation module in addition to the core, this should be announced by "CityGML [Building, Relief, Vegetation]". In case the base profile is supported, this should be indicated by "CityGML [full]".

9.1. CityGML core and extension modules

Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However, a single namespace shall not be directly included in two modules. Thus, all elements belonging to one module are associated to the module's namespace only. By this means,

module elements are guaranteed to be properly separated and distinguishable in CityGML instance documents.

Compared to CityGML versions before 1.0, the aforementioned namespace conventions introduce an extra level of complexity to data files as there is no single CityGML namespace any more. In contrast, components of different CityGML modules and, thus, of different namespaces may be arbitrarily mixed within the same CityGML instance document. Furthermore, an application might have to parse instance documents containing elements of modules which are not employed by the application itself. These parsing problems though can easily be overcome by non-“schema-aware” applications, i.e. applications that do not parse and interpret GML application schemas in a generic way. Elements from different namespaces than those declared by the application’s employed CityGML profile could be skipped. Comparable observations have to be made when using CityGML’s Application Domain Extension mechanism (cf. clause 10.13).

As for version 2.0 of the CityGML standard, there are no two thematic extension modules related by dependency. Thus, all extension modules are truly independent from each other and may be separately supported by implementations. However, the CityGML core module is a dependency for any extension module. This means that the XML schema file of the core module is imported by each XML schema file defining an extension.

The dependency relations between CityGML’s modules are illustrated in Fig. 8 using an UML package diagram. Each module is represented by a package. The package names correspond to the module names. A dashed arrow in the figure indicates that the schema at the tail of the arrow depends upon the schema at the head of the arrow. For CityGML modules, a dependency occurs where one schema <import>s another schema and accordingly the corresponding XML namespace. For example, the extension module Building imports the schema of the CityGML Core module. A short description of each module is given in Tab. 4.

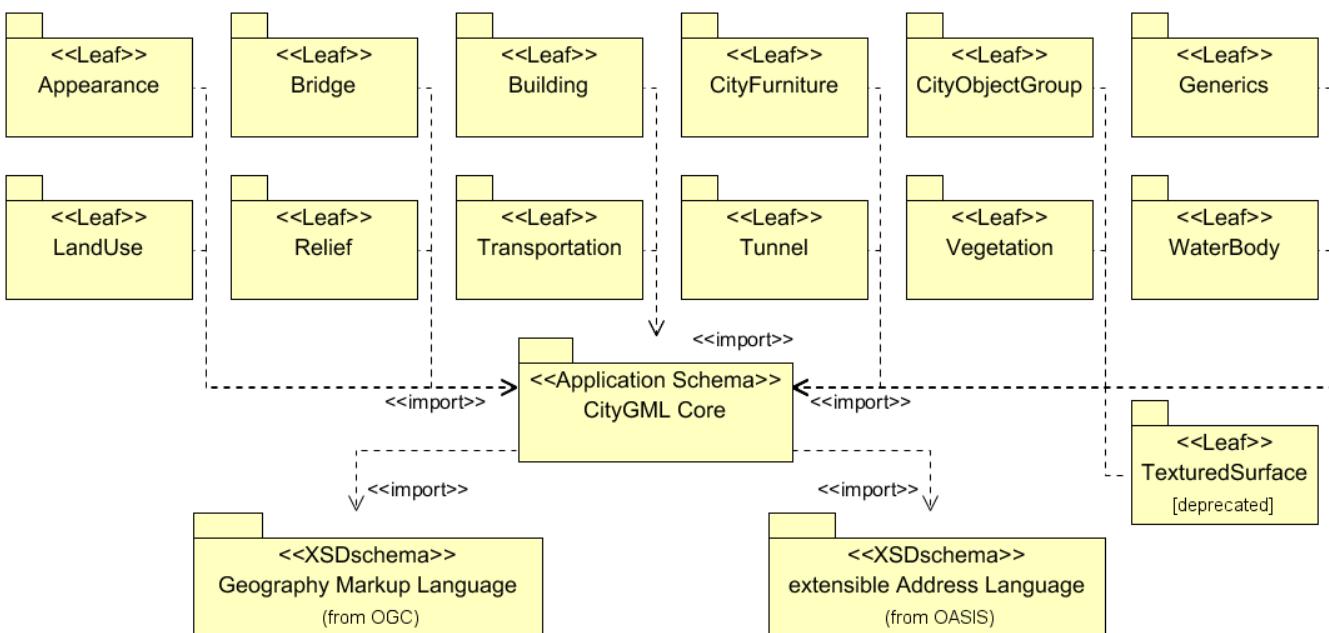


Figure 8. UML package diagram illustrating the separate modules of CityGML and their schema dependencies. Each extension module (indicated by the leaf packages) further imports the GML 3.1.1 schema definition in order to represent spatial properties of its thematic classes. For readability reasons, the corresponding dependencies have been omitted.

Module name	CityGML Core
XML namespace identifier	http://www.opengis.net/citygml/2.0
XML Schema file	cityGML.Base.xsd
Recommended namespace prefix	core
Module description	<p>The CityGML Core module defines the basic components of the CityGML data model. Primarily, this comprises abstract base classes from which all thematic classes are (transitively) derived. But also non-abstract content common to more than one extension module, for example basic data types, is defined within the core module.</p> <p>The core module itself imports the XML schema definition files of GML version 3.1.1 and the OASIS extensible Address Language xAL.</p>

Module name	Appearance
XML namespace identifier	http://www.opengis.net/citygml/appearance/2.0
XML Schema file	appearance.xsd

Recommended namespace prefix	app
Module description	The Appearance module provides the means to model appearances of CityGML features, i.e. observable properties of the feature's surface. Appearance data may be stored for each city object. Therefore, the abstract base class _CityObject defined within the core module is augmented by an additional property using CityGML's Application Domain Extension mechanism. Thus, the Appearance module has a deliberate impact on all thematic extension modules.

Module name	Bridge
XML namespace identifier	http://www.opengis.net/citygml/bridge/2.0
XML Schema file	bridge.xsd
Recommended namespace prefix	brid
Module description	The Bridge module allows the representation of thematic and spatial aspects of bridges, bridge parts, bridge installations, and interior bridge structures in four levels of detail (LOD 1 – 4).

Module name	Building
XML namespace identifier	http://www.opengis.net/citygml/building/2.0
XML Schema file	building.xsd
Recommended namespace prefix	bldg
Module description	The Building module allows the representation of thematic and spatial aspects of buildings, building parts, building installations, and interior building structures in five levels of detail (LOD 0 – 4).

Module name	CityFurniture
XML namespace identifier	http://www.opengis.net/citygml/cityfurniture/2.0
XML Schema file	cityFurniture.xsd
Recommended namespace prefix	frn
Module description	The CityFurniture module is used to represent city furniture objects in cities. City furniture objects are immovable objects like lanterns, traffic signs, advertising columns, benches, or bus stops that can be found in traffic areas, residential areas, on squares, or in built-up areas.

Module name	CityObjectGroup
XML namespace identifier	http://www.opengis.net/citygml/cityobjectgroup/2.0
XML Schema file	cityObjectGroup.xsd
Recommended namespace prefix	grp
Module description	The CityObjectGroup module provides a grouping concept for CityGML. Arbitrary city objects may be aggregated in groups according to user-defined criteria to represent and transfer these aggregations as part of the city model. A group may be further classified by specific attributes.

Module name	Generics
XML namespace identifier	http://www.opengis.net/citygml/generics/2.0
XML Schema file	generics.xsd
Recommended namespace prefix	gen
Module description	<p>The Generics module provides generic extensions to the CityGML data model that may be used to model and exchange additional attributes and features not covered by the predefined thematic classes of CityGML. However, generic extensions shall only be used if appropriate thematic classes or attributes are not provided by any other CityGML module.</p> <p>In order to represent generic attributes, the Generics module augments the abstract base class <code>_CityObject</code> defined within the core module by an additional property using CityGML's Application Domain Extension mechanism. Thus, the Generics module has a deliberate impact on all thematic extension modules.</p>

Module name	LandUse
XML namespace identifier	http://www.opengis.net/citygml/landuse/2.0
XML Schema file	landUse.xsd
Recommended namespace prefix	luse
Module description	The LandUse module allows for the representation of areas of the earth's surface dedicated to a specific land use.

Module name	Relief
-------------	--------

XML namespace identifier	http://www.opengis.net/citygml/relief/2.0
XML Schema file	relief.xsd
Recommended namespace prefix	dem
Module description	The Relief module allows for the representation of the terrain in a city model. CityGML supports terrain representations in different levels of detail, reflecting different accuracies or resolutions. The terrain may be specified as a regular raster or grid, as a TIN, by break lines, and by mass points.

Module name	Transportation
XML namespace identifier	http://www.opengis.net/citygml/transportation/2.0
XML Schema file	transportation.xsd
Recommended namespace prefix	tran
Module description	The Transportation module is used to represent the transportation features within a city, for example roads, tracks, railways, or squares. Transportation features may be represented as a linear network or by geometrically describing their 3D surfaces.

Module name	Tunnel
XML namespace identifier	http://www.opengis.net/citygml/tunnel/2.0
XML Schema file	tunnel.xsd
Recommended namespace prefix	tun
Module description	The Tunnel module facilitates the representation of thematic and spatial aspects of tunnels, tunnel parts, tunnel installations, and interior tunnel structures in four level of detail (LOD 1 – 4)

Module name	Vegetation
XML namespace identifier	http://www.opengis.net/citygml/vegetation/2.0
XML Schema file	vegetation.xsd
Recommended namespace prefix	veg

Module description	The Vegetation module provides thematic classes to represent vegetation objects. CityGML's vegetation model distinguishes between solitary vegetation objects like trees, and vegetation areas which represent biotopes like forests or other plant communities.
--------------------	--

Module name	WaterBody
XML namespace identifier	http://www.opengis.net/citygml/waterbody/2.0
XML Schema file	waterBody.xsd
Recommended namespace prefix	wtr
Module description	The WaterBody module represents the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. It does, however, not inherit any hydrological or other dynamic aspects so far.

Module name	Textured Surface [deprecated]
XML namespace identifier	http://www.opengis.net/citygml/texturesurface/2.0
XML Schema file	texturedSurface.xsd
Recommended namespace prefix	tex
Module description	The TexturedSurface module allows for assigning visual appearance properties (color, shininess, transparency) and textures to 3D surfaces. Due to inherent limitations of its modelling approach this module has been marked deprecated and is expected to be removed in future CityGML versions. Appearance information provided by this module can be converted to CityGML's Appearance module without information loss. Thus, the use of the TexturedSurface module is strongly discouraged.

9.2. CityGML profiles

A CityGML profile is a combination of thematic extension modules in conjunction with the core module of CityGML. Each CityGML instance document shall employ the CityGML profile appropriate to the provided data. In general, two approaches to employ a CityGML profile within an instance document can be differentiated:

1. CityGML profile definition embedded inline the CityGML instance document A CityGML profile can be bound to an instance document using the schemaLocation attribute defined in the XML Schema instance namespace, <http://www.w3.org/2001/XMLSchema-instance> (commonly associated with the prefix xsi). The xsi:schemaLocation attribute provides a way to locate the XML Schema definition for namespaces defined in an XML instance document. Its value is a

whitespace-delimited list of pairs of Uniform Resource Identifiers (URIs) where each pair consists of a namespace followed by the location of that namespace's XML Schema definition, which is typically a .xsd file.

By this means, the namespaces of the respective CityGML modules shall be defined within a CityGML instance document. The xsi:schemaLocation attribute then shall be used to provide the location to the respective XML Schema definition of each module. All example instance documents given in Annex G follow this first approach.

2. CityGML profile definition provided by a separate XML Schema definition file The CityGML profile may also be specified by its own XML Schema file. This schema file shall combine the appropriate CityGML modules by importing the corresponding XML Schema definitions. For this purpose, the import element defined in the XML Schema namespace shall be used, <http://www.w3.org/2001/XMLSchema> (commonly associated with the prefix xs). For the xs:import element, the namespace of the imported CityGML module along with the location of the namespace's XML Schema definition have to be declared. In order to apply a CityGML profile to an instance document, the profile's schema has to be bound to the instance document using the xsi:schemaLocation attribute. The XML Schema file of the CityGML profile shall not contain any further content.

The targetNamespace of the profile's schema shall differ from the namespaces of the imported CityGML modules. The namespace associated with the profile should be in control of the originator of the instance document and must be given as a previously unused and globally unique URI. The profile's XML Schema file must be available (or accessible on the internet) to everybody parsing the associated CityGML instance document.

The second approach is illustrated by the following example XML Schema definition for the base profile of CityGML. Since the base profile is the union of all CityGML modules, the corresponding XML Schema definition imports each and every CityGML module. By this means, all components of the CityGML data model are available in and may be exchanged by instance documents referencing this example base profile. The schema definition file of the base profile is shipped with the CityGML schema package, and is accessible at <http://schemas.opengis.net/citygml/profiles/base/2.0/CityGML.xsd>.

NOTE replace XML with UML if feasible.

The following excerpt of a CityGML dataset exemplifies how to apply the base profile schema CityGML.xsd to a CityGML instance document. The dataset contains two building objects and a city object group. The base profile defined by CityGML.xsd is referenced using the xsi:schemaLocation attribute of the root element. Thus, all CityGML modules are employed by the instance document and no further references to the XML Schema documents of the CityGML modules are necessary.

NOTE replace XML with UML if feasible

Chapter 10. Spatial Model

NOTE

This section was not generated from the CityGML Conceptual Model. TODO: identify where this info resides in the UML model, then update accordingly.

Spatial properties of CityGML features are represented by objects of GML3's geometry model. This model is based on the standard ISO 19107 'Spatial Schema' (Herring 2001), representing 3D geometry according to the well-known Boundary Representation (B-Rep, cf. Foley et al. 1995).

CityGML actually uses only a subset of the GML3 geometry package, defining a profile of GML3. This subset is depicted in Fig. 9 and Fig. 10. Further-more, GML3's explicit Boundary Representation is extended by scene graph concepts, which allow the representation of the geometry of features with the same shape implicitly and thus more space efficiently (chapter 8.2).

NOTE

Version 2.0 only provides conformance requirements for implicit geometries. Additional requirements will be needed for the other categories.

10.1. Geometric-topological model

NOTE

short intro here

10.1.1. Primitives and Composites

NOTE

short intro here

For a more detailed discussion of Primitives and Composites, see [CityGML Best Practice Section nnn](#).

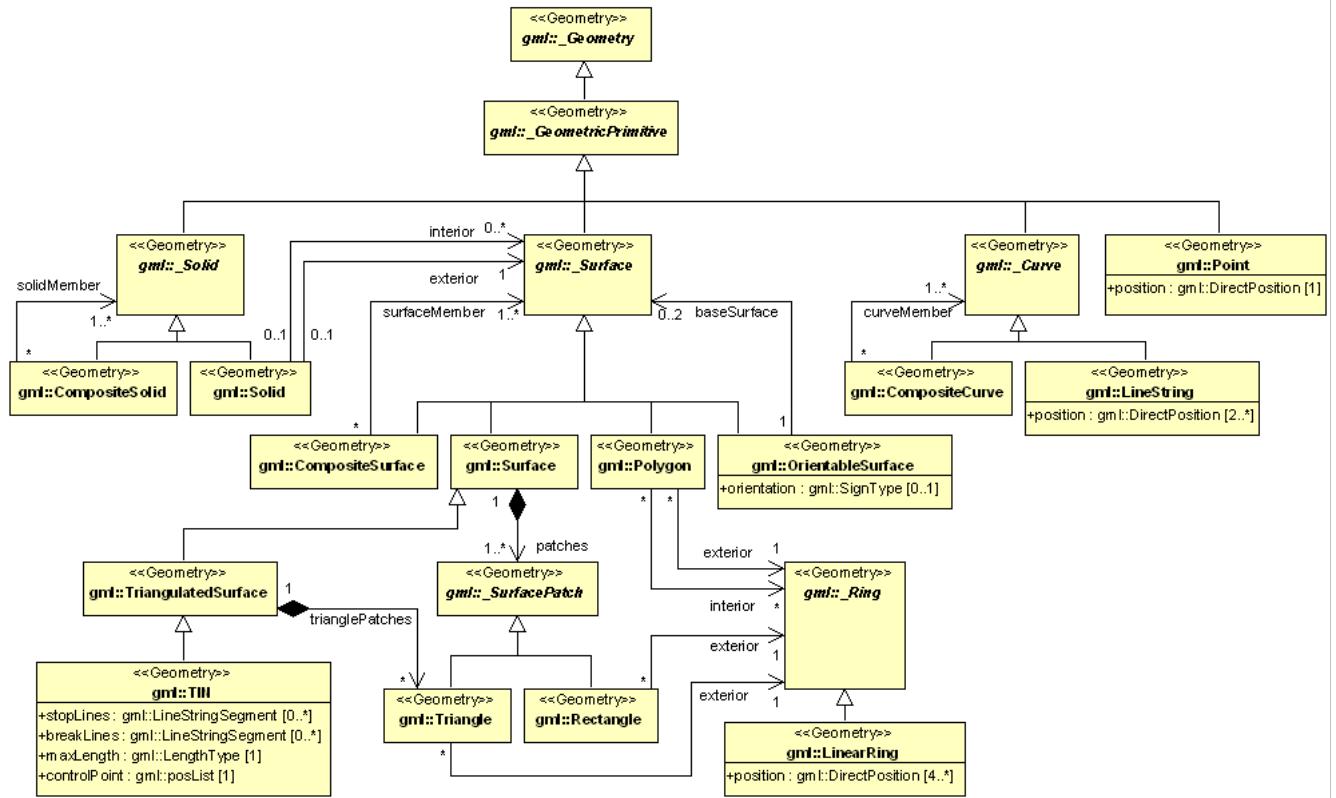


Figure 9. UML diagram of CityGML’s geometry model (subset and profile of GML3): Primitives and Composites.

10.1.2. Complexes and Aggregates

NOTE short intro here

For a more detailed discussion of Complexes and Aggregates, see [CityGML Best Practice Section nnn](#).

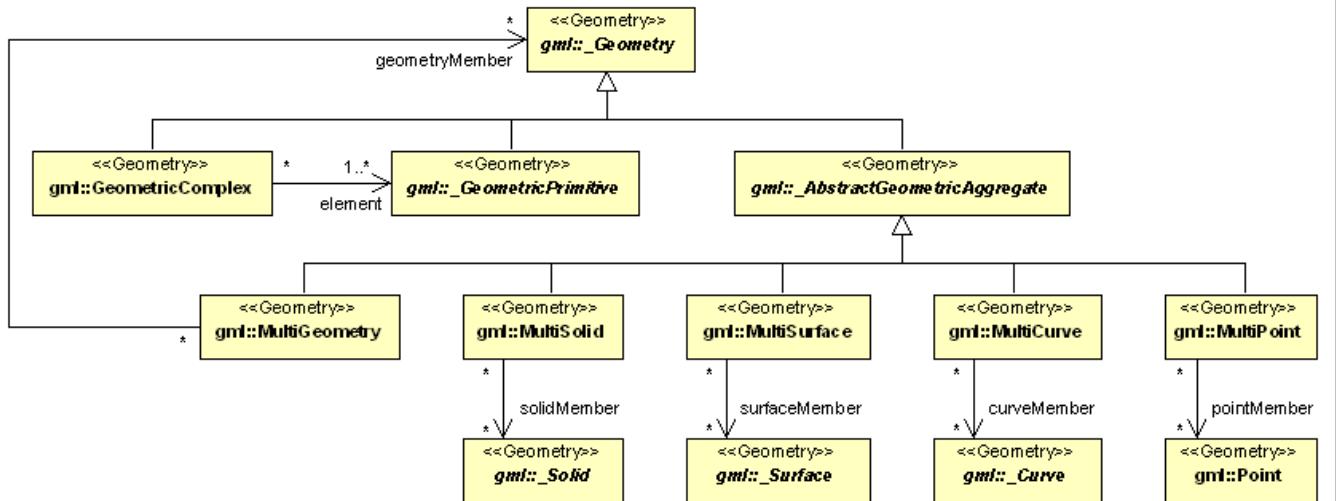


Figure 10. UML diagram of CityGML’s geometry model: Complexes and Aggregates

10.1.3. Combined Geometries

NOTE | short intro here

For a more detailed discussion of Combined Geometries, see [CityGML Best Practice Section nnn](#).

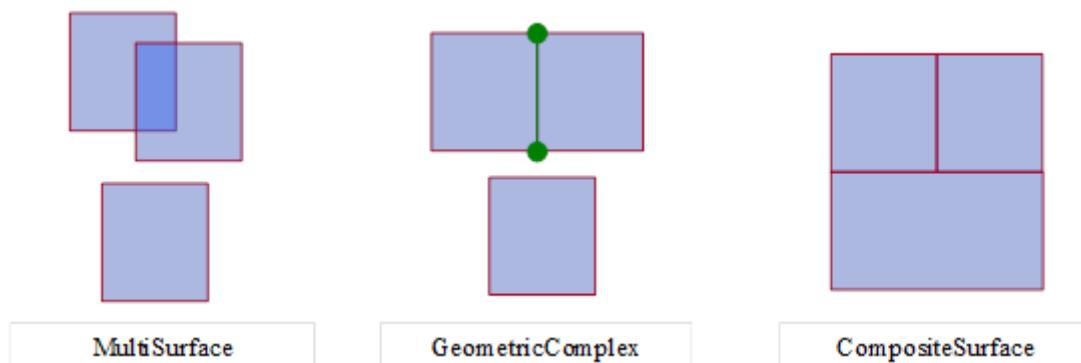


Figure 11. Combined geometries

10.1.4. Recursive Aggregation

NOTE | short intro here

For a more detailed discussion of Recursive Aggregation, see [CityGML Best Practice Section nnn](#).

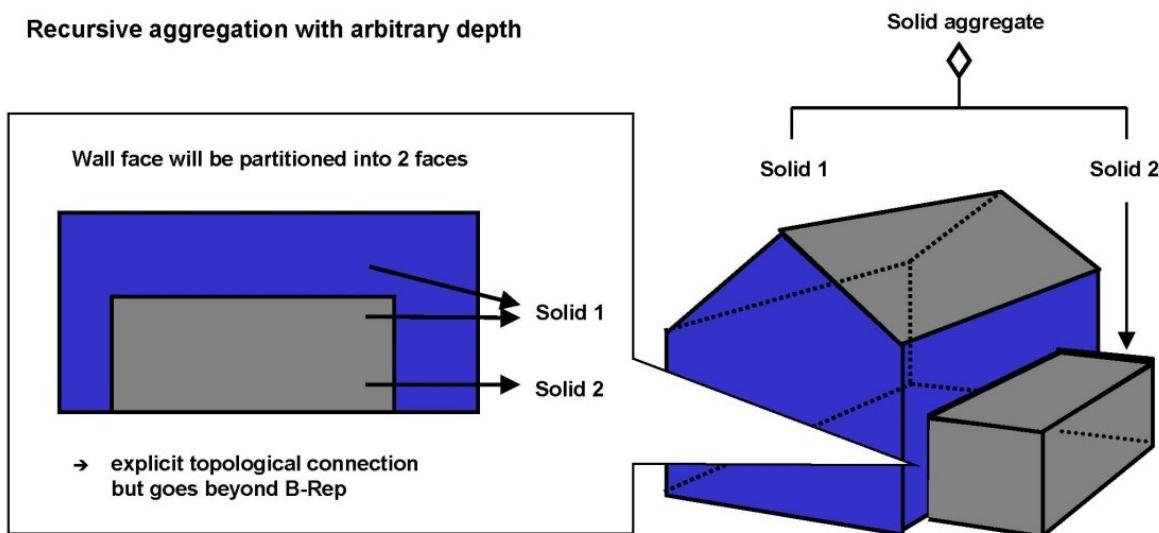


Figure 12. Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

10.2. Spatial Reference System

NOTE [short intro here](#)

For a more detailed discussion of Spatial Reference Systems, see [CityGML Best Practice Section nnn](#).

NOTE [add SRS requirements here](#)

10.3. Implicit geometries, prototypic objects, scene graph concepts

NOTE [short intro here](#)

For a more detailed discussion of Implicit Geometries, see [CityGML Best Practice Section nnn](#).

Visual Paradigm for UML Standard Edition (Technische Universität Berlin)

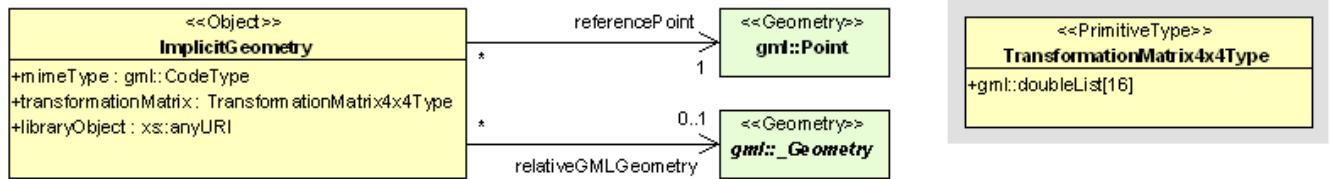


Figure 13. UML diagram of *ImplicitGeometries*. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Core module.

Chapter 11. CityGML Conceptual Model

This section provides a detailed discussion of the CityGML Conceptual Model.

Chapter 12. Appearance Model

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-appearance>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

In addition to spatial properties, CityGML features have appearances – observable properties of the feature's surface. Appearances are not limited to visual data but represent arbitrary categories called themes such as infrared radiation, noise pollution, or earthquake-induced structural stress. Each LOD can have an individual appearance for a specific theme. An appearance is composed of data for each surface geometry object, i.e. surface data. A single surface geometry object may have surface data for multiple themes. Similarly, surface data can be shared by multiple surface geometry objects (e.g. road paving). Finally, surface data values can either be constant across a surface or depend on the exact location within the surface.

CityGML's appearance model is defined within the extension module Appearance (cf. chapter 7). The UML diagram of the appearance model is illustrated in [Appearance UML Diagram](#). The Data Dictionary for the Appearance Package is provided in section [Appearance Data Dictionary](#).

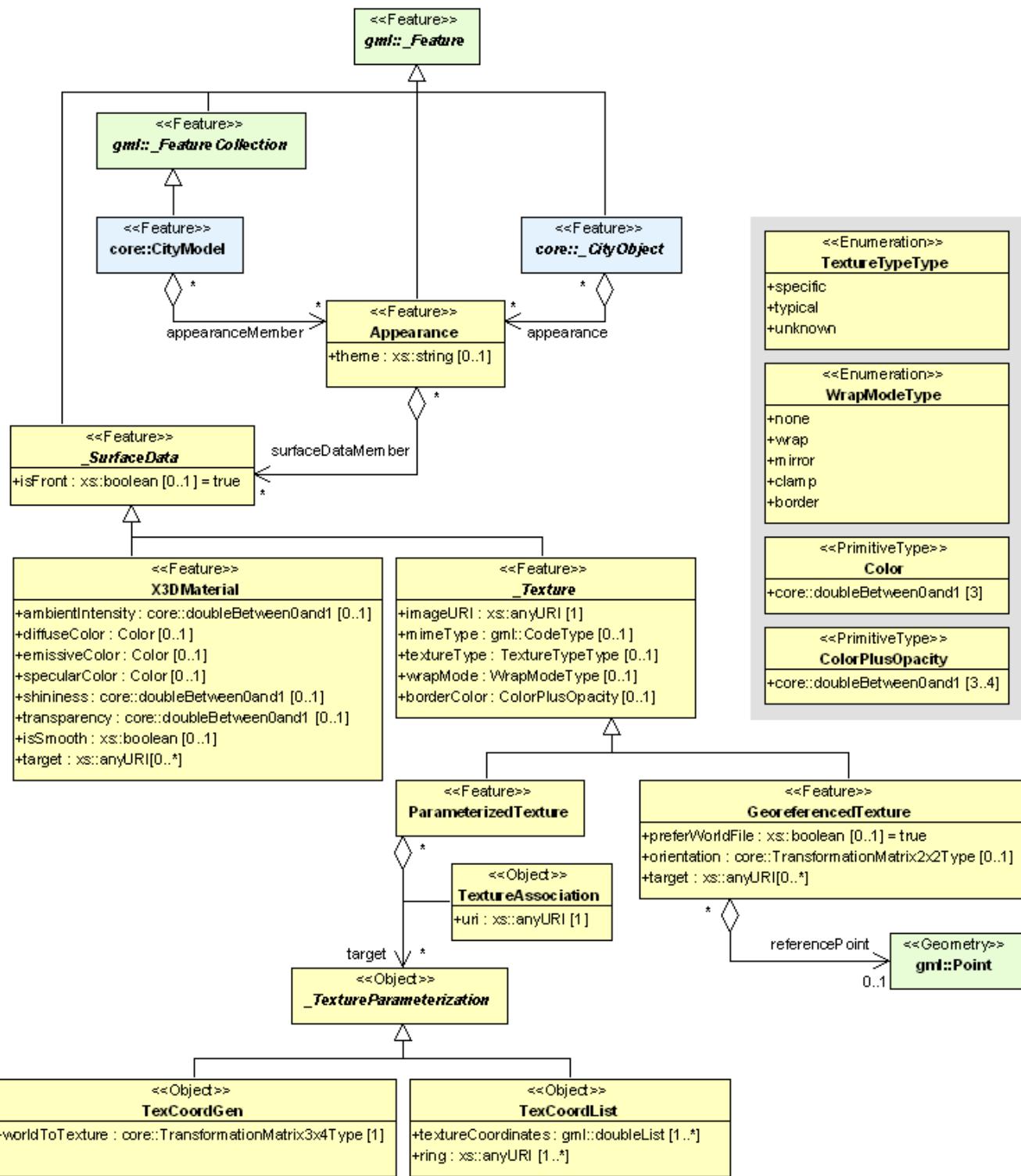


Figure 14. UML diagram of CityGML's appearance model.

The [UML diagram of CityGML's appearance model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.1. Appearance Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.1.1. Class AbstractSurfaceData

Subclass of <-- section,>>

Requirement 1	/req/Appearance/AbstractSurfaceData
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractSurfaceData UML class as documented in the AbstractSurfaceData section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractSurfaceData UML class as documented in the AbstractSurfaceData section of the Appearance Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractSurfaceData UML class; including the name, definition, type, and cardinality of those documented in the AbstractSurfaceData section of the Appearance Data Dictionary .

Class AbstractSurfaceData

Definition:

Subtype Of: <-- section,>>

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [X3DMaterial](#)

Target Role:

Target Class: [AbstractSurfaceData](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractTexture
Target Role:	
Target Class:	AbstractSurfaceData
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Appearance
Target Role:	surfaceData
Target Class:	AbstractSurfaceData
Attributes	
Attribute Name:	isFront
Value Type:	Boolean
Definition:	SIG3D: Indicates whether the X3DMaterial, GeoreferencedTexture or, ParametrizedTexture is assigned to the front side or back side of the surface
Multiplicity:	[0..1]
Stereotype:	«Property»

12.1.2. Class AbstractTexture

Subclass of [AbstractSurfaceData](#)

Requirement 2	/req/Appearance/AbstractTexture
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTexture UML class as documented in the AbstractTexture section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTexture UML class as documented in the AbstractTexture section of the Appearance Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTexture UML class; including the name, definition, type, and cardinality of those documented in the AbstractTexture section of the Appearance Data Dictionary .

Class AbstractTexture

Definition:
Subtype Of: [AbstractSurfaceData](#)
Stereotype: «FeatureType»

Associations

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [AbstractTexture](#)
Target Role:
Target Class: [AbstractSurfaceData](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [GeoreferencedTexture](#)
Target Role:
Target Class: [AbstractTexture](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [ParameterizedTexture](#)
Target Role:
Target Class: [AbstractTexture](#)

Attributes

Attribute Name: borderColor
Value Type: ColorPlusOpacity
Definition: SIG3D: Color definition for texture borders.
Multiplicity: [0..1]
Stereotype: «Property»

Attribute Name: imageURI
Value Type: URI
Definition: SIG3D: URI identifying the image data resource for the texture
Multiplicity:
Stereotype: «Property»

Attribute Name: mimeType
Value Type: Code
Definition: SIG3D: Mime type describing the data format of the image resource
Multiplicity: [0..1]
Stereotype: «Property»

Attribute Name:	textureType
Value Type:	TextureType
Definition:	"SIG3D: Distinction between prototypical (value ""typical""), object specific (value ""specific""") textures, and textures with unknown classification (value ""unknown""")."
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	wrapMode
Value Type:	WrapMode
Definition:	Collada: Definition of behavior when accessing the texture outside the underlying image raster
Multiplicity:	[0..1]
Stereotype:	«Property»

12.1.3. Class Appearance

Subclass of [AbstractAppearance](#)

Requirement 3	/req/Appearance/Appearance
A	The Implementation Specification SHALL contain an element with the same definition as the Appearance UML class as documented in the Appearance section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Appearance UML class as documented in the Appearance section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Appearance UML class; including the name, definition, type, and cardinality of those documented in the Appearance section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Appearance UML class; including the name, definition, type, and cardinality of those documented in the Appearance section of the Appearance Data Dictionary .

Class Appearance

Definition:
 Subtype Of: [AbstractAppearance](#)
 Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Appearance
Target Role:	
Target Class:	AbstractAppearance
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Appearance
Target Role:	surfaceData
Target Class:	AbstractSurfaceData
Attributes	
Attribute Name:	theme
Value Type:	CharacterString
Definition:	SIG3D: Theme name for all surfaceDataMembers. A theme is a category defining the semantics of the referenced surfaceDataMembers (e.g. infrared radiation,
Multiplicity:	[0..1]
Stereotype:	«Property»

12.1.4. Class Color

Subclass of [DoubleBetween0and1List](#)

Requirement 4	/req/Appearance/Color
A	Any use of the Color type in the Implementation Specification SHALL have the same definition as the Color UML class as documented in the Color section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Color UML class as documented in the Color section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Color UML class; including the name, definition, type, and cardinality of those documented in the Color section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Color UML class; including the name, definition, type, and cardinality of those documented in the Color section of the Appearance Data Dictionary .

Class Color

Definition:

Subtype Of: [DoubleBetween0and1List](#)

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Color](#)

Target Role:

Target Class: [DoubleBetween0and1List](#)

Name:

Type: NoteLink

Direction:

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [Color](#)

Attributes

12.1.5. Class ColorPlusOpacity

Subclass of [DoubleBetween0and1List](#)

Requirement 5	/req/Appearance/ColorPlusOpacity
A	Any use of the ColorPlusOpacity type in the Implementation Specification SHALL have the same definition as the ColorPlusOpacity UML class as documented in the ColorPlusOpacity section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ColorPlusOpacity UML class as documented in the ColorPlusOpacity section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ColorPlusOpacity UML class; including the name, definition, type, and cardinality of those documented in the ColorPlusOpacity section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ColorPlusOpacity UML class; including the name, definition, type, and cardinality of those documented in the ColorPlusOpacity section of the Appearance Data Dictionary .

Class ColorPlusOpacity

Definition:

Subtype Of: [DoubleBetween0and1List](#)

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [ColorPlusOpacity](#)

Target Role:

Target Class: [DoubleBetween0and1List](#)

Name:

Type: NoteLink

Direction:

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [ColorPlusOpacity](#)

Attributes

12.1.6. Class GeoreferencedTexture

Subclass of [AbstractTexture](#)

Requirement 6	/req/Appearance/GeoreferencedTexture
A	The Implementation Specification SHALL contain an element with the same definition as the GeoreferencedTexture UML class as documented in the GeoreferencedTexture section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GeoreferencedTexture UML class as documented in the GeoreferencedTexture section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GeoreferencedTexture UML class; including the name, definition, type, and cardinality of those documented in the GeoreferencedTexture section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GeoreferencedTexture UML class; including the name, definition, type, and cardinality of those documented in the GeoreferencedTexture section of the Appearance Data Dictionary .

Class GeoreferencedTexture

Definition:

Subtype Of: [AbstractTexture](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [GeoreferencedTexture](#)

Target Role: referencePoint

Target Class: [GM_Point](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [GeoreferencedTexture](#)

Target Role:

Target Class: [AbstractTexture](#)

Attributes

Attribute Name: orientation

Value Type: TransformationMatrix2x2

Definition: SIG3D: Defines the rotation and scaling of the image in form of a 2x2 matrix (a list of 4 doubles in row-major order).

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: preferWorldFile

Value Type: Boolean

Definition: SIG3D: Flag for defining the precedence of an accompanying worldfile before the georeference included in the image source. If this value is false, the world file

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: target

Value Type: URI

Definition: SIG3D: Geometry object which is associated with the texture.

Multiplicity: [0..*]

Stereotype: «Property»

12.1.7. Class ParameterizedTexture

Subclass of [AbstractTexture](#)

Requirement 7	/req/Appearance/ParameterizedTexture
---------------	--------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the ParameterizedTexture UML class as documented in the ParameterizedTexture section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ParameterizedTexture UML class as documented in the ParameterizedTexture section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ParameterizedTexture UML class; including the name, definition, type, and cardinality of those documented in the ParameterizedTexture section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ParameterizedTexture UML class; including the name, definition, type, and cardinality of those documented in the ParameterizedTexture section of the Appearance Data Dictionary .

Class ParameterizedTexture

Definition:

Subtype Of: [AbstractTexture](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [ParameterizedTexture](#)

Target Role:

Target Class: [AbstractTexture](#)

Name:

Type: AssociationClass

Direction: Source → Destination

Source Role:

Source Class: [ParameterizedTexture](#)

Target Role: textureParameterization

Target Class: [AbstractTextureParameterization](#)

Attributes

12.1.8. Class TextureAssociation

Subclass of <-- section,>

Requirement 8	/req/Appearance/TextureAssociation
A	The Implementation Specification SHALL contain an element with the same definition as the TextureAssociation UML class as documented in the TextureAssociation section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TextureAssociation UML class as documented in the TextureAssociation section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TextureAssociation UML class; including the name, definition, type, and cardinality of those documented in the TextureAssociation section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TextureAssociation UML class; including the name, definition, type, and cardinality of those documented in the TextureAssociation section of the Appearance Data Dictionary .

Class TextureAssociation

Definition:

Subtype Of: <<section,>>

Stereotype: «ObjectType»

Associations

Attributes

Attribute Name: uri

Value Type: URI

Definition: SIG3D: Link to the geometry object to be textured.

Multiplicity:

Stereotype: «Property»

12.1.9. Class X3DMaterial

Subclass of [AbstractSurfaceData](#)

Requirement 9	/req/Appearance/X3DMaterial
A	The Implementation Specification SHALL contain an element with the same definition as the X3DMaterial UML class as documented in the X3DMaterial section of the Appearance Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the X3DMaterial UML class as documented in the X3DMaterial section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the X3DMaterial UML class; including the name, definition, type, and cardinality of those documented in the X3DMaterial section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the X3DMaterial UML class; including the name, definition, type, and cardinality of those documented in the X3DMaterial section of the Appearance Data Dictionary .

Class X3DMaterial

Definition:

Subtype Of: [AbstractSurfaceData](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [X3DMaterial](#)

Target Role:

Target Class: [AbstractSurfaceData](#)

Attributes

Attribute Name: ambientIntensity

Value Type: DoubleBetween0and1

Definition: X3D: Minimum percentage of diffuseColor that is visible regardless of light sources

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: diffuseColor

Value Type: Color

Definition: X3D: Color of the diffusely reflected light

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: emissiveColor

Value Type: Color

Definition: X3D: Color of the light emitted by the surface

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	isSmooth
Value Type:	Boolean
Definition:	X3D: Hint for normal interpolation. If true vertex normals used for shading. Otherwise normals are constant for the patch.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	shininess
Value Type:	DoubleBetween0and1
Definition:	X3D: Controls the sharpness of specular highlights. 0 produces a soft glow. 1 produces sharp highlights.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	specularColor
Value Type:	Color
Definition:	X3D: Color of the directly reflected light
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	target
Value Type:	URI
Definition:	X3D: URI identifying the target surface geometry to apply the material properties
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	transparency
Value Type:	DoubleBetween0and1
Definition:	X3D: Degree of transparency of the surface. 0 means fully opaque. 1 means fully transparent.
Multiplicity:	[0..1]
Stereotype:	«Property»

12.1.10. Class AbstractTextureParameterization

Subclass of <– section,>>

Requirement 10	/req/Appearance/AbstractTextureParameterization
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTextureParameterization UML class as documented in the AbstractTextureParameterization section of the Appearance Data Dictionary .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTextureParameterization UML class as documented in the AbstractTextureParameterization section of the Appearance Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTextureParameterization UML class; including the name, definition, type, and cardinality of those documented in the AbstractTextureParameterization section of the Appearance Data Dictionary .

Class AbstractTextureParameterization

Definition:

Subtype Of: <-> section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TexCoordGen](#)

Target Role:

Target Class: [AbstractTextureParameterization](#)

Name:

Type: AssociationClass

Direction: Source → Destination

Source Role:

Source Class: [ParameterizedTexture](#)

Target Role: textureParameterization

Target Class: [AbstractTextureParameterization](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TexCoordList](#)

Target Role:

Target Class: [AbstractTextureParameterization](#)

Attributes

12.1.11. Class TexCoordGen

Subclass of [AbstractTextureParameterization](#)

Requirement 11	/req/Appearance/TexCoordGen
A	Any use of the TexCoordGen type in the Implementation Specification SHALL have the same definition as the TexCoordGen UML class as documented in the TexCoordGen section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TexCoordGen UML class as documented in the TexCoordGen section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TexCoordGen UML class; including the name, definition, type, and cardinality of those documented in the TexCoordGen section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TexCoordGen UML class; including the name, definition, type, and cardinality of those documented in the TexCoordGen section of the Appearance Data Dictionary .

Class TexCoordGen

Definition:

Subtype Of: <->section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TexCoordGen](#)

Target Role:

Target Class: [AbstractTextureParameterization](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TexCoordGen](#)

Target Role: crs

Target Class: [SC_CRS](#)

Attributes

Attribute Name:	worldToTexture
Value Type:	TransformationMatrix3x4
Definition:	SIG3D: 3x4 Matrix that defines the transformation between world coordinates and texture coordinates.
Multiplicity:	
Stereotype:	«Property»

12.1.12. Class TexCoordList

Subclass of [AbstractTextureParameterization](#)

Requirement 12	/req/Appearance/TexCoordList
A	Any use of the TexCoordList type in the Implementation Specification SHALL have the same definition as the TexCoordList UML class as documented in the TexCoordList section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TexCoordList UML class as documented in the TexCoordList section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TexCoordList UML class; including the name, definition, type, and cardinality of those documented in the TexCoordList section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TexCoordList UML class; including the name, definition, type, and cardinality of those documented in the TexCoordList section of the Appearance Data Dictionary .

Class TexCoordList

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «DataType»

Associations

Name:
 Type: Generalization
 Direction: Source → Destination
 Source Role:
 Source Class: [TexCoordList](#)
 Target Role:
 Target Class: [AbstractTextureParameterization](#)

Attributes

Attribute Name:	ring
Value Type:	URI
Definition:	SIG3D: gml:ids of the LinearRings that are parameterized using the given texture coordinates
Multiplicity:	[1..*]
Stereotype:	«Property»
Attribute Name:	textureCoordinates
Value Type:	DoubleList
Definition:	SIG3D: List of texture coordinates with two doubles per ring vertex
Multiplicity:	[1..*]
Stereotype:	«Property»

12.1.13. Class TextureType

Subclass of <-- section,>>

Requirement 13	/req/Appearance/TextureType
A	Any use of the TextureType type in the Implementation Specification SHALL have the same definition as the TextureType UML class as documented in the TextureType section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TextureType UML class as documented in the TextureType section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TextureType UML class; including the name, definition, type, and cardinality of those documented in the TextureType section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TextureType UML class; including the name, definition, type, and cardinality of those documented in the TextureType section of the Appearance Data Dictionary .

Class TextureType

Definition:
 Subtype Of: <-- section,>>
 Stereotype:

Associations

Attributes

Attribute Name:	specific
Value Type:	
Definition:	SIG3D: The texture is specific for a certain object
Multiplicity:	
Stereotype:	
Attribute Name:	typical
Value Type:	
Definition:	SIG3D: The texture is typical for a kind of object
Multiplicity:	
Stereotype:	
Attribute Name:	unknown
Value Type:	
Definition:	SIG3D: The texture type is unknown.
Multiplicity:	
Stereotype:	

12.1.14. Class WrapMode

Subclass of <-- section,>>

Requirement 14	/req/Appearance/WrapMode
A	Any use of the WrapMode type in the Implementation Specification SHALL have the same definition as the WrapMode UML class as documented in the WrapMode section of the Appearance Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WrapMode UML class as documented in the WrapMode section of the Appearance Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the WrapMode UML class; including the name, definition, type, and cardinality of those documented in the WrapMode section of the Appearance Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WrapMode UML class; including the name, definition, type, and cardinality of those documented in the WrapMode section of the Appearance Data Dictionary .

Class WrapMode
Definition:
Subtype Of: <-- section,>>
Stereotype:
Associations

Attributes

Attribute Name: none

Value Type:

Definition: COLLADA: The resulting color is fully transparent

Multiplicity:

Stereotype:

Attribute Name: wrap

Value Type:

Definition: COLLADA: The texture is repeated

Multiplicity:

Stereotype:

Attribute Name: mirror

Value Type:

Definition: COLLADA: The texture is repeated and mirrored

Multiplicity:

Stereotype:

Attribute Name: clamp

Value Type:

Definition: COLLADA: The texture is clamped to its edges

Multiplicity:

Stereotype:

Attribute Name: border

Value Type:

Definition: COLLADA: The resulting color is specified by the borderColor element (RGBA)

Multiplicity:

Stereotype:

12.1.15. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.2. Core

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-core>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

The CityGML Core module defines the basic concepts and components of the overall CityGML data model. It forms the universal lower bound of the CityGML data model and, thus, is a dependency of all extension modules. Consequently, the core module has to be implemented by any conformant system. Primarily, the core module provides the abstract base classes from which thematic classes within extension modules are (transitively) derived. Besides abstract type definitions, the core also contains non-abstract content, for example basic data types and thematic classes that may be used by more than one extension module. The UML diagram in Fig. 21 illustrates CityGML's core module, for the XML Schema definition see below and annex A.1.

The UML diagram of the CityGML Core is depicted in [Core UML Diagram](#). The Data Dictionary for the Core Package is provided in section [Core Data Dictionary](#).

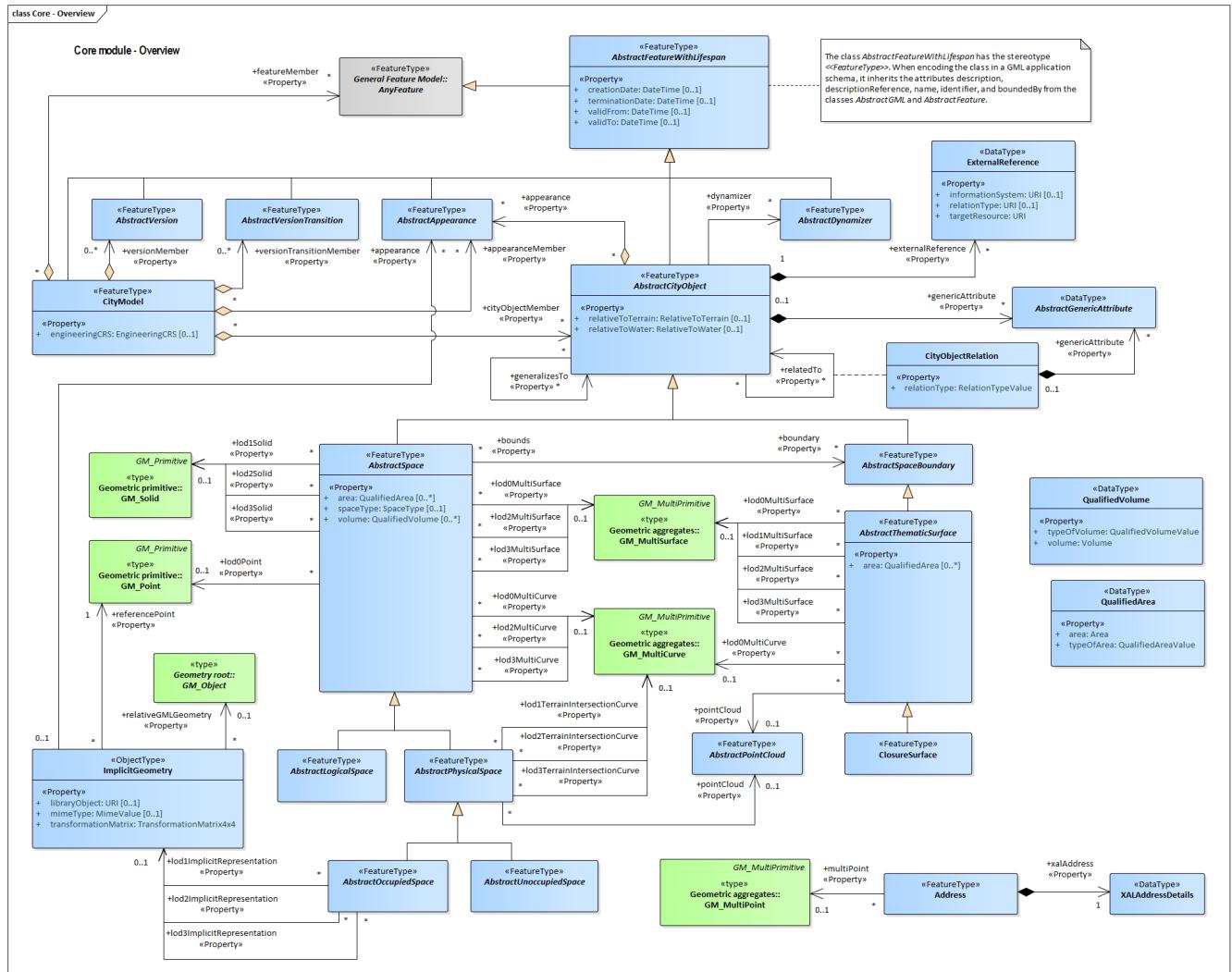


Figure 15. UML diagram of CityGML's core module.

The [UML diagram of CityGML's core module](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.3. Core Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.3.1. Class AbstractAppearance

Subclass of [AbstractFeatureWithLifespan](#)

Requirement 15	/req/Core/AbstractAppearance
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractAppearance UML class as documented in the AbstractAppearance section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractAppearance UML class as documented in the AbstractAppearance section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractAppearance UML class; including the name, definition, type, and cardinality of those documented in the AbstractAppearance section of the Core Data Dictionary .

Class AbstractAppearance

Definition:

Subtype Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractAppearance](#)

Target Role:

Target Class: [AbstractFeatureWithLifespan](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractCityObject
Target Role:	appearance
Target Class:	AbstractAppearance
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	CityModel
Target Role:	appearanceMember
Target Class:	AbstractAppearance
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	ImplicitGeometry
Target Role:	appearance
Target Class:	AbstractAppearance
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Appearance
Target Role:	
Target Class:	AbstractAppearance
Attributes	

12.3.2. Class [AbstractCityObject](#)

Subclass of [AbstractFeatureWithLifespan](#)

Requirement 16	/req/Core/AbstractCityObject
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractCityObject UML class as documented in the AbstractCityObject section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractCityObject UML class as documented in the AbstractCityObject section of the Core Data Dictionary .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractCityObject UML class; including the name, definition, type, and cardinality of those documented in the AbstractCityObject section of the Core Data Dictionary .
---	--

Class AbstractCityObject

Definition:

Subtype Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractCityObject](#)

Target Role: appearance

Target Class: [AbstractAppearance](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractCityObject](#)

Target Role: externalReference

Target Class: [ExternalReference](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractCityObject](#)

Target Role: dynamizer

Target Class: [AbstractDynamizer](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractCityObject](#)

Target Role: generalizesTo

Target Class: [AbstractCityObject](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractCityObject</p> <p>Target Role: genericAttribute</p> <p>Target Class: AbstractGenericAttribute</p>
<p>Name:</p> <p>Type: AssociationClass</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractCityObject</p> <p>Target Role: relatedTo</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractCityObject</p> <p>Target Role:</p> <p>Target Class: AbstractFeatureWithLifespan</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: CityModel</p> <p>Target Role: cityObjectMember</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: AssociationClass</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: CityObjectGroup</p> <p>Target Role: groupMember</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: SensorConnection</p> <p>Target Role: sensorLocation</p> <p>Target Class: AbstractCityObject</p>

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: CityObjectGroup</p> <p>Target Role: parent</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpaceBoundary</p> <p>Target Role:</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractCityObject</p> <p>Target Role: generalizesTo</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role:</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: AssociationClass</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractCityObject</p> <p>Target Role: relatedTo</p> <p>Target Class: AbstractCityObject</p>
<p>Attributes</p> <p>Attribute Name: relativeToTerrain</p> <p>Value Type: RelativeToTerrain</p> <p>Definition: SIG3D: Vertical position of the AbstractCityObject relative to the surrounding terrain.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

Attribute Name:	relativeToWater
Value Type:	RelativeToWater
Definition:	SIG3D: Vertical position of the AbstractCityObject relative to a surrounding water surface.
Multiplicity:	[0..1]
Stereotype:	«Property»

12.3.3. Class AbstractDynamizer

Subclass of [AbstractFeatureWithLifespan](#)

Requirement 17	/req/Core/AbstractDynamizer
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractDynamizer UML class as documented in the AbstractDynamizer section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractDynamizer UML class as documented in the AbstractDynamizer section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractDynamizer UML class; including the name, definition, type, and cardinality of those documented in the AbstractDynamizer section of the Core Data Dictionary .

Class AbstractDynamizer

Definition:
 Subtype Of: [AbstractFeatureWithLifespan](#)
 Stereotype: «FeatureType»

Associations

Name:
 Type: Generalization
 Direction: Source → Destination
 Source Role:
 Source Class: [AbstractDynamizer](#)
 Target Role:
 Target Class: [AbstractFeatureWithLifespan](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractCityObject
Target Role:	dynamizer
Target Class:	AbstractDynamizer
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Dynamizer
Target Role:	
Target Class:	AbstractDynamizer
Attributes	

12.3.4. Class AbstractFeatureWithLifespan

Subclass of [AnyFeature](#)

Requirement 18	/req/Core/AbstractFeatureWithLifespan
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFeatureWithLifespan UML class as documented in the AbstractFeatureWithLifespan section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFeatureWithLifespan UML class as documented in the AbstractFeatureWithLifespan section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFeatureWithLifespan UML class; including the name, definition, type, and cardinality of those documented in the AbstractFeatureWithLifespan section of the Core Data Dictionary .

Class AbstractFeatureWithLifespan

Definition:

Subtype Of: [AnyFeature](#)

Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractFeatureWithLifespan
Target Role:	
Target Class:	AnyFeature
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractVersion
Target Role:	
Target Class:	AbstractFeatureWithLifespan
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Transaction
Target Role:	oldFeature
Target Class:	AbstractFeatureWithLifespan
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractAppearance
Target Role:	
Target Class:	AbstractFeatureWithLifespan
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Version
Target Role:	versionMember
Target Class:	AbstractFeatureWithLifespan
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractVersionTransition
Target Role:	
Target Class:	AbstractFeatureWithLifespan

<p>Name:</p> <p>Type: NoteLink</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Note</p> <p>Target Role:</p> <p>Target Class: AbstractFeatureWithLifespan</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: CityModel</p> <p>Target Role:</p> <p>Target Class: AbstractFeatureWithLifespan</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Transaction</p> <p>Target Role: newFeature</p> <p>Target Class: AbstractFeatureWithLifespan</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractDynamizer</p> <p>Target Role:</p> <p>Target Class: AbstractFeatureWithLifespan</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractCityObject</p> <p>Target Role:</p> <p>Target Class: AbstractFeatureWithLifespan</p>
<p>Attributes</p> <p>Attribute Name: creationDate</p> <p>Value Type: DateTime</p> <p>Definition:</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

Attribute Name: terminationDate

Value Type: DateTime

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: validFrom

Value Type: DateTime

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: validTo

Value Type: DateTime

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

12.3.5. Class AbstractLogicalSpace

Subclass of [AbstractSpace](#)

Requirement 19	/req/Core/AbstractLogicalSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractLogicalSpace UML class as documented in the AbstractLogicalSpace section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractLogicalSpace UML class as documented in the AbstractLogicalSpace section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractLogicalSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractLogicalSpace section of the Core Data Dictionary .

Class AbstractLogicalSpace

Definition:

Subtype Of: [AbstractSpace](#)

Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractLogicalSpace
Target Role:	
Target Class:	AbstractSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	CityObjectGroup
Target Role:	
Target Class:	AbstractLogicalSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GenericLogicalSpace
Target Role:	
Target Class:	AbstractLogicalSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBuildingSubdivision
Target Role:	
Target Class:	AbstractLogicalSpace
Attributes	

12.3.6. Class **AbstractOccupiedSpace**

Subclass of [AbstractPhysicalSpace](#)

Requirement 20	/req/Core/AbstractOccupiedSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractOccupiedSpace UML class as documented in the AbstractOccupiedSpace section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractOccupiedSpace UML class as documented in the AbstractOccupiedSpace section of the Core Data Dictionary .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractOccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractOccupiedSpace section of the Core Data Dictionary .
---	--

Class AbstractOccupiedSpace

Definition:

Subtype Of: [AbstractPhysicalSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractOccupiedSpace](#)

Target Role: lod2ImplicitRepresentation

Target Class: [ImplicitGeometry](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractOccupiedSpace](#)

Target Role:

Target Class: [AbstractPhysicalSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractOccupiedSpace](#)

Target Role: lod1ImplicitRepresentation

Target Class: [ImplicitGeometry](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractOccupiedSpace](#)

Target Role: lod3ImplicitRepresentation

Target Class: [ImplicitGeometry](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractFurniture
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	WaterBody
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GenericOccupiedSpace
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractConstructiveElement
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractVegetationObject
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractInstallation
Target Role:	
Target Class:	AbstractOccupiedSpace

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	CityFurniture
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractConstruction
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractFillingElement
Target Role:	
Target Class:	AbstractOccupiedSpace
Attributes	

12.3.7. Class AbstractPhysicalSpace

Subclass of [AbstractSpace](#)

Requirement 21	/req/Core/AbstractPhysicalSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractPhysicalSpace UML class as documented in the AbstractPhysicalSpace section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractPhysicalSpace UML class as documented in the AbstractPhysicalSpace section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractPhysicalSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractPhysicalSpace section of the Core Data Dictionary .

Class AbstractPhysicalSpace

Definition:
Subtype Of: [AbstractSpace](#)
StereoType: «FeatureType»

Associations

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [AbstractPhysicalSpace](#)
Target Role:
Target Class: [AbstractSpace](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractPhysicalSpace](#)
Target Role: pointCloud
Target Class: [AbstractPointCloud](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractPhysicalSpace](#)
Target Role: lod3TerrainIntersectionCurve
Target Class: [GM_MultiCurve](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractPhysicalSpace](#)
Target Role: lod2TerrainIntersectionCurve
Target Class: [GM_MultiCurve](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractPhysicalSpace](#)
Target Role: lod1TerrainIntersectionCurve
Target Class: [GM_MultiCurve](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractUnoccupiedSpace
Target Role:	
Target Class:	AbstractPhysicalSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractOccupiedSpace
Target Role:	
Target Class:	AbstractPhysicalSpace
Attributes	

12.3.8. Class AbstractPointCloud

Subclass of <-- section,>>

Requirement 22	/req/Core/AbstractPointCloud
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractPointCloud UML class as documented in the AbstractPointCloud section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractPointCloud UML class as documented in the AbstractPointCloud section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractPointCloud UML class; including the name, definition, type, and cardinality of those documented in the AbstractPointCloud section of the Core Data Dictionary .

Class AbstractPointCloud
Definition:
Subtype Of: <-- section,>>
Stereotype: «FeatureType»
Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	PointCloud
Target Role:	
Target Class:	AbstractPointCloud
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	MassPointRelief
Target Role:	pointCloud
Target Class:	AbstractPointCloud
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractPhysicalSpace
Target Role:	pointCloud
Target Class:	AbstractPointCloud
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractThematicSurface
Target Role:	pointCloud
Target Class:	AbstractPointCloud
Attributes	

12.3.9. Class AbstractSpace

Subclass of [AbstractCityObject](#)

Requirement 23	/req/Core/AbstractSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractSpace UML class as documented in the AbstractSpace section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractSpace UML class as documented in the AbstractSpace section of the Core Data Dictionary .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractSpace section of the Core Data Dictionary .
---	--

Class AbstractSpace

Definition:

Subtype Of: [AbstractCityObject](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractSpace](#)

Target Role: lod3MultiCurve

Target Class: [GM_MultiCurve](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractSpace](#)

Target Role: lod2MultiSurface

Target Class: [GM_MultiSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractSpace](#)

Target Role: lod3Solid

Target Class: [GM_Solid](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractSpace](#)

Target Role: lod2Solid

Target Class: [GM_Solid](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role: lod1Solid</p> <p>Target Class: GM_Solid</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role: lod0Point</p> <p>Target Class: GM_Point</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role: bounds</p> <p>Source Class: AbstractSpace</p> <p>Target Role: boundary</p> <p>Target Class: AbstractSpaceBoundary</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role: lod0MultiSurface</p> <p>Target Class: GM_MultiSurface</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role: lod2MultiCurve</p> <p>Target Class: GM_MultiCurve</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role: lod3MultiSurface</p> <p>Target Class: GM_MultiSurface</p>

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role: lod0MultiCurve</p> <p>Target Class: GM_MultiCurve</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractSpace</p> <p>Target Role:</p> <p>Target Class: AbstractCityObject</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractPhysicalSpace</p> <p>Target Role:</p> <p>Target Class: AbstractSpace</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractLogicalSpace</p> <p>Target Role:</p> <p>Target Class: AbstractSpace</p>
<p>Attributes</p> <p>Attribute Name: area</p> <p>Value Type: QualifiedArea</p> <p>Definition:</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p> <p>Attribute Name: spaceType</p> <p>Value Type: SpaceType</p> <p>Definition: Degree of openness of an abstract space.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p> <p>Attribute Name: volume</p> <p>Value Type: QualifiedVolume</p> <p>Definition:</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

12.3.10. Class AbstractSpaceBoundary

Subclass of [AbstractCityObject](#)

Requirement 24	/req/Core/AbstractSpaceBoundary
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractSpaceBoundary UML class as documented in the AbstractSpaceBoundary section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractSpaceBoundary UML class as documented in the AbstractSpaceBoundary section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractSpaceBoundary UML class; including the name, definition, type, and cardinality of those documented in the AbstractSpaceBoundary section of the Core Data Dictionary .

Class AbstractSpaceBoundary

Definition:

Subtype Of: [AbstractCityObject](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractSpaceBoundary](#)

Target Role:

Target Class: [AbstractCityObject](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractThematicSurface](#)

Target Role:

Target Class: [AbstractSpaceBoundary](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	bounds
Source Class:	AbstractSpace
Target Role:	boundary
Target Class:	AbstractSpaceBoundary
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	ReliefFeature
Target Role:	
Target Class:	AbstractSpaceBoundary
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractReliefComponent
Target Role:	
Target Class:	AbstractSpaceBoundary
Attributes	

12.3.11. Class AbstractThematicSurface

Subclass of [AbstractSpaceBoundary](#)

Requirement 25	/req/Core/AbstractThematicSurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractThematicSurface UML class as documented in the AbstractThematicSurface section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractThematicSurface UML class as documented in the AbstractThematicSurface section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractThematicSurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractThematicSurface section of the Core Data Dictionary .

Class AbstractThematicSurface

Definition:
Subtype Of: [AbstractSpaceBoundary](#)
StereoType: «FeatureType»

Associations

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractThematicSurface](#)
Target Role: lod1MultiSurface
Target Class: [GM_MultiSurface](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [AbstractThematicSurface](#)
Target Role:
Target Class: [AbstractSpaceBoundary](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractThematicSurface](#)
Target Role: lod0MultiSurface
Target Class: [GM_MultiSurface](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractThematicSurface](#)
Target Role: lod2MultiSurface
Target Class: [GM_MultiSurface](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractThematicSurface](#)
Target Role: lod3MultiSurface
Target Class: [GM_MultiSurface](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractThematicSurface
Target Role:	lod0MultiCurve
Target Class:	GM_MultiCurve
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractThematicSurface
Target Role:	pointCloud
Target Class:	AbstractPointCloud
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TrafficArea
Target Role:	
Target Class:	AbstractThematicSurface
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractInstallation
Target Role:	boundary
Target Class:	AbstractThematicSurface
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	BuildingRoom
Target Role:	boundary
Target Class:	AbstractThematicSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractConstructionSurface
Target Role:	
Target Class:	AbstractThematicSurface

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Storey</p> <p>Target Role: boundary</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractWaterBoundarySurface</p> <p>Target Role:</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Marking</p> <p>Target Role:</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractConstruction</p> <p>Target Role: boundary</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: LandUse</p> <p>Target Role:</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BridgeRoom</p> <p>Target Role: boundary</p> <p>Target Class: AbstractThematicSurface</p>

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractConstructiveElement</p> <p>Target Role: boundary</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: HoleSurface</p> <p>Target Role:</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractFillingSurface</p> <p>Target Role:</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Hole</p> <p>Target Role: boundary</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: HollowSpace</p> <p>Target Role: boundary</p> <p>Target Class: AbstractThematicSurface</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: ClosureSurface</p> <p>Target Role:</p> <p>Target Class: AbstractThematicSurface</p>

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GenericThematicSurface
Target Role:	
Target Class:	AbstractThematicSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AuxiliaryTrafficArea
Target Role:	
Target Class:	AbstractThematicSurface
Attributes	
Attribute Name:	area
Value Type:	QualifiedArea
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.3.12. Class AbstractUnoccupiedSpace

Subclass of [AbstractPhysicalSpace](#)

Requirement 26	/req/Core/AbstractUnoccupiedSpace
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractUnoccupiedSpace UML class as documented in the AbstractUnoccupiedSpace section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractUnoccupiedSpace UML class as documented in the AbstractUnoccupiedSpace section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractUnoccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractUnoccupiedSpace section of the Core Data Dictionary .

Class AbstractUnoccupiedSpace

Definition:
Subtype Of: [AbstractPhysicalSpace](#)
StereoType: «FeatureType»

Associations

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [AbstractUnoccupiedSpace](#)
Target Role:
Target Class: [AbstractPhysicalSpace](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [GenericUnoccupiedSpace](#)
Target Role:
Target Class: [AbstractUnoccupiedSpace](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [AbstractTransportationSpace](#)
Target Role:
Target Class: [AbstractUnoccupiedSpace](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [BuildingRoom](#)
Target Role:
Target Class: [AbstractUnoccupiedSpace](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [BridgeRoom](#)
Target Role:
Target Class: [AbstractUnoccupiedSpace](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TrafficSpace
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AuxiliaryTrafficSpace
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Hole
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	HollowSpace
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	ClearanceSpace
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Attributes	

12.3.13. Class AbstractVersion

Subclass of [AbstractFeatureWithLifespan](#)

Requirement 27	/req/Core/AbstractVersion
-----------------------	---------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractVersion UML class as documented in the AbstractVersion section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractVersion UML class as documented in the AbstractVersion section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractVersion UML class; including the name, definition, type, and cardinality of those documented in the AbstractVersion section of the Core Data Dictionary .

Class AbstractVersion

Definition:

Subtype Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractVersion](#)

Target Role:

Target Class: [AbstractFeatureWithLifespan](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityModel](#)

Target Role: versionMember

Target Class: [AbstractVersion](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Version](#)

Target Role:

Target Class: [AbstractVersion](#)

Attributes

12.3.14. Class AbstractVersionTransition

Subclass of [AbstractFeatureWithLifespan](#)

Requirement 28	/req/Core/AbstractVersionTransition
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractVersionTransition UML class as documented in the AbstractVersionTransition section of the Core Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractVersionTransition UML class as documented in the AbstractVersionTransition section of the Core Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractVersionTransition UML class; including the name, definition, type, and cardinality of those documented in the AbstractVersionTransition section of the Core Data Dictionary .

Class AbstractVersionTransition

Definition:

Subtype Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractVersionTransition](#)

Target Role:

Target Class: [AbstractFeatureWithLifespan](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityModel](#)

Target Role: versionTransitionMember

Target Class: [AbstractVersionTransition](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	VersionTransition
Target Role:	
Target Class:	AbstractVersionTransition
Attributes	

12.3.15. Class Address

Subclass of <-- section,>>

Requirement 29	/req/Core/Address
A	The Implementation Specification SHALL contain an element with the same definition as the Address UML class as documented in the Address section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Address UML class as documented in the Address section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Address UML class; including the name, definition, type, and cardinality of those documented in the Address section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Address UML class; including the name, definition, type, and cardinality of those documented in the Address section of the Core Data Dictionary .

Class Address
Definition:
Subtype Of: <-- section,>>
Stereotype: «FeatureType»
Associations
Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: Address
Target Role: xalAddress
Target Class: XALAddressDetails

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Address</p> <p>Target Role: multiPoint</p> <p>Target Class: GM_MultiPoint</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: DoorSurface</p> <p>Target Role: address</p> <p>Target Class: Address</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBridge</p> <p>Target Role: address</p> <p>Target Class: Address</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuilding</p> <p>Target Role: address</p> <p>Target Class: Address</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BuildingUnit</p> <p>Target Role: address</p> <p>Target Class: Address</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Door</p> <p>Target Role: address</p> <p>Target Class: Address</p>
Attributes

12.3.16. Class CityModel

Subclass of [AbstractFeatureWithLifespan](#)

Requirement 30	/req/Core/CityModel
A	The Implementation Specification SHALL contain an element with the same definition as the CityModel UML class as documented in the CityModel section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityModel UML class as documented in the CityModel section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the CityModel UML class; including the name, definition, type, and cardinality of those documented in the CityModel section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityModel UML class; including the name, definition, type, and cardinality of those documented in the CityModel section of the Core Data Dictionary .

Class CityModel

Definition:

Subtype Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityModel](#)

Target Role: cityObjectMember

Target Class: [AbstractCityObject](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityModel](#)

Target Role: versionMember

Target Class: [AbstractVersion](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	CityModel
Target Role:	versionTransitionMember
Target Class:	AbstractVersionTransition
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	CityModel
Target Role:	appearanceMember
Target Class:	AbstractAppearance
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	CityModel
Target Role:	featureMember
Target Class:	AnyFeature
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	CityModel
Target Role:	
Target Class:	AbstractFeatureWithLifespan
Attributes	
Attribute Name:	engineeringCRS
Value Type:	EngineeringCRS
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.3.17. Class CityObjectRelation

Subclass of <– section,>>

Requirement 31	/req/Core/CityObjectRelation
A	Any use of the CityObjectRelation type in the Implementation Specification SHALL have the same definition as the CityObjectRelation UML class as documented in the CityObjectRelation section of the Core Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityObjectRelation UML class as documented in the CityObjectRelation section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the CityObjectRelation UML class; including the name, definition, type, and cardinality of those documented in the CityObjectRelation section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityObjectRelation UML class; including the name, definition, type, and cardinality of those documented in the CityObjectRelation section of the Core Data Dictionary .

Class CityObjectRelation

Definition:

Subtype Of: <-> section,>>

Stereotype:

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityObjectRelation](#)

Target Role: genericAttribute

Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: relationType

Value Type: RelationTypeValue

Definition:

Multiplicity:

Stereotype: «Property»

12.3.18. Class ClosureSurface

Subclass of [AbstractThematicSurface](#)

Requirement 32	/req/Core/ClosureSurface
A	The Implementation Specification SHALL contain an element with the same definition as the ClosureSurface UML class as documented in the ClosureSurface section of the Core Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ClosureSurface UML class as documented in the ClosureSurface section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ClosureSurface UML class; including the name, definition, type, and cardinality of those documented in the ClosureSurface section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ClosureSurface UML class; including the name, definition, type, and cardinality of those documented in the ClosureSurface section of the Core Data Dictionary .

Class ClosureSurface

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [ClosureSurface](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Attributes

12.3.19. Class DoubleBetween0and1

Subclass of <-- section,>>

Requirement 33	/req/Core/DoubleBetween0and1
A	Any use of the DoubleBetween0and1 type in the Implementation Specification SHALL have the same definition as the DoubleBetween0and1 UML class as documented in the DoubleBetween0and1 section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleBetween0and1 UML class as documented in the DoubleBetween0and1 section of the Core Data Dictionary .

C	The implementation Specification SHALL represent the attributes of the DoubleBetween0and1 UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1 section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleBetween0and1 UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1 section of the Core Data Dictionary .

Class DoubleBetween0and1

Definition:

Subtype Of: <-- section,>>

Stereotype: «BasicType»

Associations

Name:

Type: NoteLink

Direction:

Source Role:

Source Class: Note

Target Role:

Target Class: DoubleBetween0and1

Attributes

12.3.20. Class DoubleBetween0and1List

Subclass of <-- section,>>

Requirement 34	/req/Core/DoubleBetween0and1List
A	Any use of the DoubleBetween0and1List type in the Implementation Specification SHALL have the same definition as the DoubleBetween0and1List UML class as documented in the DoubleBetween0and1List section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleBetween0and1List UML class as documented in the DoubleBetween0and1List section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the DoubleBetween0and1List UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1List section of the Core Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleBetween0and1List UML class; including the name, definition, type, and cardinality of those documented in the DoubleBetween0and1List section of the Core Data Dictionary .
---	--

Class DoubleBetween0and1List

Definition:

Subtype Of: <-- section,>>

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Color](#)

Target Role:

Target Class: [DoubleBetween0and1List](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [ColorPlusOpacity](#)

Target Role:

Target Class: [DoubleBetween0and1List](#)

Attributes

Attribute Name: list

Value Type: DoubleBetween0and1

Definition:

Multiplicity:

Stereotype:

12.3.21. Class DoubleList

Subclass of <-- section,>>

Requirement 35	/req/Core/DoubleList
A	Any use of the DoubleList type in the Implementation Specification SHALL have the same definition as the DoubleList UML class as documented in the DoubleList section of the Core Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleList UML class as documented in the DoubleList section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the DoubleList UML class; including the name, definition, type, and cardinality of those documented in the DoubleList section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleList UML class; including the name, definition, type, and cardinality of those documented in the DoubleList section of the Core Data Dictionary .

Class DoubleList

Definition:

Subtype Of: <-> section,>>

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TransformationMatrix4x4](#)

Target Role:

Target Class: [DoubleList](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TransformationMatrix3x4](#)

Target Role:

Target Class: [DoubleList](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TransformationMatrix2x2](#)

Target Role:

Target Class: [DoubleList](#)

Attributes

Attribute Name: list

Value Type: Real

Definition:

Multiplicity:

Stereotype:

12.3.22. Class ImplicitGeometry

Subclass of <-- section,>>

Requirement 36	/req/Core/ImplicitGeometry
A	The Implementation Specification SHALL contain an element with the same definition as the ImplicitGeometry UML class as documented in the ImplicitGeometry section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ImplicitGeometry UML class as documented in the ImplicitGeometry section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ImplicitGeometry UML class; including the name, definition, type, and cardinality of those documented in the ImplicitGeometry section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ImplicitGeometry UML class; including the name, definition, type, and cardinality of those documented in the ImplicitGeometry section of the Core Data Dictionary .

Class ImplicitGeometry

Definition:

Subtype Of: <-- section,>>

Stereotype: «ObjectType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [ImplicitGeometry](#)

Target Role: appearance

Target Class: [AbstractAppearance](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: ImplicitGeometry</p> <p>Target Role: relativeGMLGeometry</p> <p>Target Class: GM_Object</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: ImplicitGeometry</p> <p>Target Role: referencePoint</p> <p>Target Class: GM_Point</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractOccupiedSpace</p> <p>Target Role: lod2ImplicitRepresentation</p> <p>Target Class: ImplicitGeometry</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractOccupiedSpace</p> <p>Target Role: lod1ImplicitRepresentation</p> <p>Target Class: ImplicitGeometry</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractOccupiedSpace</p> <p>Target Role: lod3ImplicitRepresentation</p> <p>Target Class: ImplicitGeometry</p>
<p>Attributes</p> <p>Attribute Name: libraryObject</p> <p>Value Type: URI</p> <p>Definition: SIG3D: External link to a prototype geometry.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

Attribute Name:	mimeType
Value Type:	MimeType
Definition:	SIG3D: Mime type of the referenced external geometric object (attribute libraryObject).
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	transformationMatrix
Value Type:	TransformationMatrix4x4
Definition:	SIG3D: Mathematical transformation (translation, rotation and scaling) between the prototype geometry and the actual spatial position of the object.
Multiplicity:	
Stereotype:	«Property»

12.3.23. Class IntegerBetween0and3

Subclass of <-- section,>>

Requirement 37	/req/Core/IntegerBetween0and3
A	Any use of the IntegerBetween0and3 type in the Implementation Specification SHALL have the same definition as the IntegerBetween0and3 UML class as documented in the IntegerBetween0and3 section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the IntegerBetween0and3 UML class as documented in the IntegerBetween0and3 section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the IntegerBetween0and3 UML class; including the name, definition, type, and cardinality of those documented in the IntegerBetween0and3 section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the IntegerBetween0and3 UML class; including the name, definition, type, and cardinality of those documented in the IntegerBetween0and3 section of the Core Data Dictionary .

Class IntegerBetween0and3

Definition:	
Subtype Of:	<-- section,>>
Stereotype:	«BasicType»

Associations

Name:	
Type:	NoteLink
Direction:	
Source Role:	
Source Class:	Note
Target Role:	
Target Class:	IntegerBetween0and3
Attributes	

12.3.24. Class IntervalValue

Subclass of <-- section,>>

Requirement 38	/req/Core/IntervalValue
A	Any use of the IntervalValue type in an Implementation Specification SHALL be restricted to the valid values specified in the IntervalValue UML class as documented in the IntervalValue section of the Core Data Dictionary .

Class IntervalValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»
Associations
Name:
Type: NoteLink
Direction: Source → Destination
Source Role:
Source Class: Note
Target Role:
Target Class: IntervalValue
Attributes

12.3.25. Class MimeValue

Subclass of <-- section,>>

Requirement 39	/req/Core/MimeValue
A	Any use of the MimeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the MimeValue UML class as documented in the MimeValue section of the Core Data Dictionary .

Class MimeValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.3.26. Class OccupantTypeValue

Subclass of <-- section,>>

Requirement 40	/req/Core/OccupantTypeValue
A	Any use of the OccupantTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OccupantTypeValue UML class as documented in the OccupantTypeValue section of the Core Data Dictionary .

Class OccupantTypeValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [OccupantTypeValue](#)

Attributes

12.3.27. Class OtherRelationTypeValue

Subclass of [RelationTypeValue](#)

Requirement 41	/req/Core/OtherRelationTypeValue
A	Any use of the OtherRelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherRelationTypeValue UML class as documented in the OtherRelationTypeValue section of the Core Data Dictionary .

Class OtherRelationTypeValue

Definition:

Subtype Of: [RelationTypeValue](#)

Stereotype: «CodeList»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [OtherRelationTypeValue](#)

Target Role:

Target Class: [RelationTypeValue](#)

Attributes

12.3.28. Class QualifiedAreaValue

Subclass of <-- section,>>

Requirement 42	/req/Core/QualifiedAreaValue
A	Any use of the QualifiedAreaValue type in an Implementation Specification SHALL be restricted to the valid values specified in the QualifiedAreaValue UML class as documented in the QualifiedAreaValue section of the Core Data Dictionary .

Class QualifiedAreaValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [QualifiedAreaValue](#)

Attributes

12.3.29. Class QualifiedVolumeValue

Subclass of <-- section,>>

Requirement 43	/req/Core/QualifiedVolumeValue
A	Any use of the QualifiedVolumeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the QualifiedVolumeValue UML class as documented in the QualifiedVolumeValue section of the Core Data Dictionary .

Class QualifiedVolumeValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: Note

Target Role:

Target Class: QualifiedVolumeValue

Attributes

12.3.30. Class RelationTypeValue

Subclass of <-- section,>>

Requirement 44	/req/Core/RelationTypeValue
A	Any use of the RelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RelationTypeValue UML class as documented in the RelationTypeValue section of the Core Data Dictionary .

Class RelationTypeValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: TemporalRelationTypeValue

Target Role:

Target Class: RelationTypeValue

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TopologicRelationTypeValue
Target Role:	
Target Class:	RelationTypeValue
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	OtherRelationTypeValue
Target Role:	
Target Class:	RelationTypeValue
Attributes	

12.3.31. Class TemporalRelationTypeValue

Subclass of [RelationTypeValue](#)

Requirement 45	/req/Core/TemporalRelationTypeValue
A	Any use of the TemporalRelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TemporalRelationTypeValue UML class as documented in the TemporalRelationTypeValue section of the Core Data Dictionary .

Class TemporalRelationTypeValue	
Definition:	
Subtype Of: RelationTypeValue	
Stereotype: «CodeList»	
Associations	
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TemporalRelationTypeValue
Target Role:	
Target Class:	RelationTypeValue
Attributes	

12.3.32. Class TopologicRelationTypeValue

Subclass of [RelationTypeValue](#)

Requirement 46	/req/Core/TopologicalRelationTypeValue
A	Any use of the TopologicalRelationTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TopologicalRelationTypeValue UML class as documented in the TopologicalRelationTypeValue section of the Core Data Dictionary .

Class TopologicRelationTypeValue

Definition:

Subtype Of: [RelationTypeValue](#)

Stereotype: «CodeList»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TopologicRelationTypeValue](#)

Target Role:

Target Class: [RelationTypeValue](#)

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [TopologicRelationTypeValue](#)

Attributes

12.3.33. Class TransformationMatrix2x2

Subclass of [DoubleList](#)

Requirement 47	/req/Core/TransformationMatrix2x2
A	Any use of the TransformationMatrix2x2 type in the Implementation Specification SHALL have the same definition as the TransformationMatrix2x2 UML class as documented in the TransformationMatrix2x2 section of the Core Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TransformationMatrix2x2 UML class as documented in the TransformationMatrix2x2 section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TransformationMatrix2x2 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix2x2 section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TransformationMatrix2x2 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix2x2 section of the Core Data Dictionary .

Class TransformationMatrix2x2

Definition:

Subtype Of: [DoubleList](#)

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TransformationMatrix2x2](#)

Target Role:

Target Class: [DoubleList](#)

Name:

Type: NoteLink

Direction:

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [TransformationMatrix2x2](#)

Attributes

12.3.34. Class TransformationMatrix3x4

Subclass of [DoubleList](#)

Requirement 48	/req/Core/TransformationMatrix3x4
----------------	-----------------------------------

A	Any use of the TransformationMatrix3x4 type in the Implementation Specification SHALL have the same definition as the TransformationMatrix3x4 UML class as documented in the TransformationMatrix3x4 section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TransformationMatrix3x4 UML class as documented in the TransformationMatrix3x4 section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TransformationMatrix3x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix3x4 section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TransformationMatrix3x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix3x4 section of the Core Data Dictionary .

Class TransformationMatrix3x4

Definition:

Subtype Of: [DoubleList](#)

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TransformationMatrix3x4](#)

Target Role:

Target Class: [DoubleList](#)

Name:

Type: NoteLink

Direction:

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [TransformationMatrix3x4](#)

Attributes

12.3.35. Class TransformationMatrix4x4

Subclass of [DoubleList](#)

Requirement 49	/req/Core/TransformationMatrix4x4
A	Any use of the TransformationMatrix4x4 type in the Implementation Specification SHALL have the same definition as the TransformationMatrix4x4 UML class as documented in the TransformationMatrix4x4 section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TransformationMatrix4x4 UML class as documented in the TransformationMatrix4x4 section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TransformationMatrix4x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix4x4 section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TransformationMatrix4x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix4x4 section of the Core Data Dictionary .

Class TransformationMatrix4x4

Definition:

Subtype Of: [DoubleList](#)

Stereotype: «BasicType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TransformationMatrix4x4](#)

Target Role:

Target Class: [DoubleList](#)

Name:

Type: NoteLink

Direction:

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [TransformationMatrix4x4](#)

Attributes

12.3.36. Class AbstractGenericAttribute

Subclass of <-- section,>>

Requirement 50	/req/Core/AbstractGenericAttribute
A	Any use of the AbstractGenericAttribute type in the Implementation Specification SHALL have the same definition as the AbstractGenericAttribute UML class as documented in the AbstractGenericAttribute section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the AbstractGenericAttribute UML class as documented in the AbstractGenericAttribute section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the AbstractGenericAttribute UML class; including the name, definition, type, and cardinality of those documented in the AbstractGenericAttribute section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the AbstractGenericAttribute UML class; including the name, definition, type, and cardinality of those documented in the AbstractGenericAttribute section of the Core Data Dictionary .

Class AbstractGenericAttribute

Definition:

Subtype Of: <<section,>>

Stereotype: «DataType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityObjectRelation](#)

Target Role: genericAttribute

Target Class: [AbstractGenericAttribute](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [GenericAttributeSet](#)

Target Role:

Target Class: [AbstractGenericAttribute](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	StringAttribute
Target Role:	
Target Class:	AbstractGenericAttribute
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	UriAttribute
Target Role:	
Target Class:	AbstractGenericAttribute
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	GenericAttributeSet
Target Role:	genericAttribute
Target Class:	AbstractGenericAttribute
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	MeasureAttribute
Target Role:	
Target Class:	AbstractGenericAttribute
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	DoubleAttribute
Target Role:	
Target Class:	AbstractGenericAttribute
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	IntAttribute
Target Role:	
Target Class:	AbstractGenericAttribute

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractCityObject
Target Role:	genericAttribute
Target Class:	AbstractGenericAttribute
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	DateAttribute
Target Role:	
Target Class:	AbstractGenericAttribute
Attributes	

12.3.37. Class ExternalReference

Subclass of <-->

Requirement 51	/req/Core/ExternalReference
A	Any use of the ExternalReference type in the Implementation Specification SHALL have the same definition as the ExternalReference UML class as documented in the ExternalReference section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ExternalReference UML class as documented in the ExternalReference section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ExternalReference UML class; including the name, definition, type, and cardinality of those documented in the ExternalReference section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ExternalReference UML class; including the name, definition, type, and cardinality of those documented in the ExternalReference section of the Core Data Dictionary .

Class ExternalReference
Definition:
Subtype Of: <-->
Stereotype: «DataType»

Associations	
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractCityObject
Target Role:	externalReference
Target Class:	ExternalReference
Attributes	
Attribute Name:	informationSystem
Value Type:	URI
Definition:	SIG3D: URL or URN of the information system or data set.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	relationType
Value Type:	URI
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	targetResource
Value Type:	URI
Definition:	SIG3D: Referenced object in the external information system or data set.
Multiplicity:	
Stereotype:	«Property»

12.3.38. Class Occupancy

Subclass of <-- section,>>

Requirement 52	/req/Core/Occupancy
A	Any use of the Occupancy type in the Implementation Specification SHALL have the same definition as the Occupancy UML class as documented in the Occupancy section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Occupancy UML class as documented in the Occupancy section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Occupancy UML class; including the name, definition, type, and cardinality of those documented in the Occupancy section of the Core Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the Occupancy UML class; including the name, definition, type, and cardinality of those documented in the Occupancy section of the Core Data Dictionary .
---	--

Class Occupancy

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Attributes

Attribute Name: interval

Value Type: IntervalValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: numberOfOccupants

Value Type: Integer

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: occupantType

Value Type: OccupantTypeValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

12.3.39. Class QualifiedArea

Subclass of <-- section,>>

Requirement 53	/req/Core/QualifiedArea
A	Any use of the QualifiedArea type in the Implementation Specification SHALL have the same definition as the QualifiedArea UML class as documented in the QualifiedArea section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the QualifiedArea UML class as documented in the QualifiedArea section of the Core Data Dictionary .

C	The implementation Specification SHALL represent the attributes of the QualifiedArea UML class; including the name, definition, type, and cardinality of those documented in the QualifiedArea section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the QualifiedArea UML class; including the name, definition, type, and cardinality of those documented in the QualifiedArea section of the Core Data Dictionary .

Class QualifiedArea

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Attributes

Attribute Name: area

Value Type: Area

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: typeOfArea

Value Type: QualifiedAreaValue

Definition:

Multiplicity:

Stereotype: «Property»

12.3.40. Class QualifiedVolume

Subclass of <-- section,>>

Requirement 54	/req/Core/QualifiedVolume
A	Any use of the QualifiedVolume type in the Implementation Specification SHALL have the same definition as the QualifiedVolume UML class as documented in the QualifiedVolume section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the QualifiedVolume UML class as documented in the QualifiedVolume section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the QualifiedVolume UML class; including the name, definition, type, and cardinality of those documented in the QualifiedVolume section of the Core Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the QualifiedVolume UML class; including the name, definition, type, and cardinality of those documented in the QualifiedVolume section of the Core Data Dictionary .
---	--

Class QualifiedVolume

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Attributes

Attribute Name: typeOfVolume

Value Type: QualifiedVolumeValue

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: volume

Value Type: Volume

Definition:

Multiplicity:

Stereotype: «Property»

12.3.41. Class RelativeToTerrain

Subclass of <-- section,>>

Requirement 55	/req/Core/RelativeToTerrain
A	Any use of the RelativeToTerrain type in an Implementation Specification SHALL be restricted to the valid values specified in the RelativeToTerrain UML class as documented in the RelativeToTerrain section of the Core Data Dictionary .

Class RelativeToTerrain

Definition:

Subtype Of: <-- section,>>

Stereotype:

Associations

Attributes

Attribute Name: entirelyAboveTerrain

Value Type:

Definition: SIG3D: Object is located entirely above terrain.

Multiplicity:

Stereotype:

Attribute Name: substantiallyAboveTerrain

Value Type:

Definition: SIG3D: Most, but not all parts of the object are located above terrain.

Multiplicity:

Stereotype:

Attribute Name: substantiallyAboveAndBelowTerrain

Value Type:

Definition: SIG3D: Parts of the object are located above terrain, and other parts below terrain.

Multiplicity:

Stereotype:

Attribute Name: substantiallyBelowTerrain

Value Type:

Definition: SIG3D: Most, but not all parts of the object are located below terrain.

Multiplicity:

Stereotype:

Attribute Name: entirelyBelowTerrain

Value Type:

Definition: SIG3D: All parts of the object are located below terrain.

Multiplicity:

Stereotype:

12.3.42. Class RelativeToWater

Subclass of <-- section,>>

Requirement 56	/req/Core/RelativeToWater
A	Any use of the RelativeToWater type in an Implementation Specification SHALL be restricted to the valid values specified in the RelativeToWater UML class as documented in the Core Data Dictionary .

Class RelativeToWater

Definition:

Subtype Of: <-- section,>>

Stereotype:

Associations

Attributes

Attribute Name:	entirelyAboveWaterSurface
Value Type:	
Definition:	SIG3D: Object is located entirely above water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	substantiallyAboveWaterSurface
Value Type:	
Definition:	SIG3D: Most, but not all parts of the object are located above water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	substantiallyAboveAndBelowWaterSurface
Value Type:	
Definition:	SIG3D: Parts of the object are located above water surface, and other parts below water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	substantiallyBelowWaterSurface
Value Type:	
Definition:	SIG3D: Most, but not all parts of the object are located below water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	entirelyBelowWaterSurface
Value Type:	
Definition:	SIG3D: All parts of the object are located below water surface.
Multiplicity:	
Stereotype:	
Attribute Name:	temporarilyAboveAndBelowWaterSurface
Value Type:	
Definition:	SIG3D: The height of the water surface is varying and the object temporarily is located above or below water level.
Multiplicity:	
Stereotype:	

12.3.43. Class SpaceType

Subclass of <-- section,>>

Requirement 57	/req/Core/SpaceType
A	Any use of the SpaceType type in an Implementation Specification SHALL be restricted to the valid values specified in the SpaceType UML class as documented in the SpaceType section of the Core Data Dictionary .

Class SpaceType

Definition:
Subtype Of: <-- section,>>
Stereotype:
Associations
Attributes
Attribute Name: closed
Value Type:
Definition: boundaries at all sides
Multiplicity:
Stereotype:
Attribute Name: open
Value Type:
Definition: boundary at the bottom only
Multiplicity:
Stereotype:
Attribute Name: semiOpen
Value Type:
Definition: at least one side has a boundary
Multiplicity:
Stereotype:

12.3.44. Class XALAddressDetails

Subclass of <-- section,>>

Requirement 58	/req/Core/XALAddressDetails
A	Any use of the XALAddressDetails type in the Implementation Specification SHALL have the same definition as the XALAddressDetails UML class as documented in the XALAddressDetails section of the Core Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the XALAddressDetails UML class as documented in the XALAddressDetails section of the Core Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the XALAddressDetails UML class; including the name, definition, type, and cardinality of those documented in the XALAddressDetails section of the Core Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the XALAddressDetails UML class; including the name, definition, type, and cardinality of those documented in the XALAddressDetails section of the Core Data Dictionary .

Class XALAddressDetails

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Address](#)

Target Role: xalAddress

Target Class: [XALAddressDetails](#)

Attributes

12.3.45. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.4. Digital Terrain Model

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-relief>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

An essential part of a city model is the terrain. The Digital Terrain Model (DTM) of CityGML is provided by the thematic extension module Relief (cf. chapter 7). In CityGML, the terrain is represented by the class ReliefFeature in LOD 0-4 (Fig. 24 depicts the UML diagram, for the XML schema definition see annex A.9). A ReliefFeature consists of one or more entities of the class ReliefComponent. Its validity may be restricted to a certain area defined by an optional validity extent polygon. As ReliefFeature and ReliefComponent are derivatives of _CityObject, the corresponding attributes and relations are inherited. The class ReliefFeature is associated with different concepts of terrain representations which can coexist. The terrain may be specified as a regular raster or grid (RasterRelief), as a TIN (Triangulated Irregular Network, TINRelief), by break lines (BreaklineRelief), or by mass points (MasspointRelief). The four types are implemented by the corresponding GML3 classes: grids by `gml:RectifiedGridCoverage`, break lines by `gml:MultiCurve`, mass points by `gml:MultiPoint` and TINs either by `gml:TriangulatedSurface` or by `gml:Tin`. In case of `gml:TriangulatedSurfaces`, the triangles are given explicitly while in case of `gml:Tin` only 3D points

are represented, where the triangulation can be reconstructed by standard methods (Delaunay triangulation, cf. Okabe et al. 1992). Break lines are represented by 3D curves. Mass points are simply a set of 3D points.

The UML diagram of the Relief Package is depicted in [Relief UML Diagram](#). The Data Dictionary for the Relief Package is provided in section [Relief Data Dictionary](#).

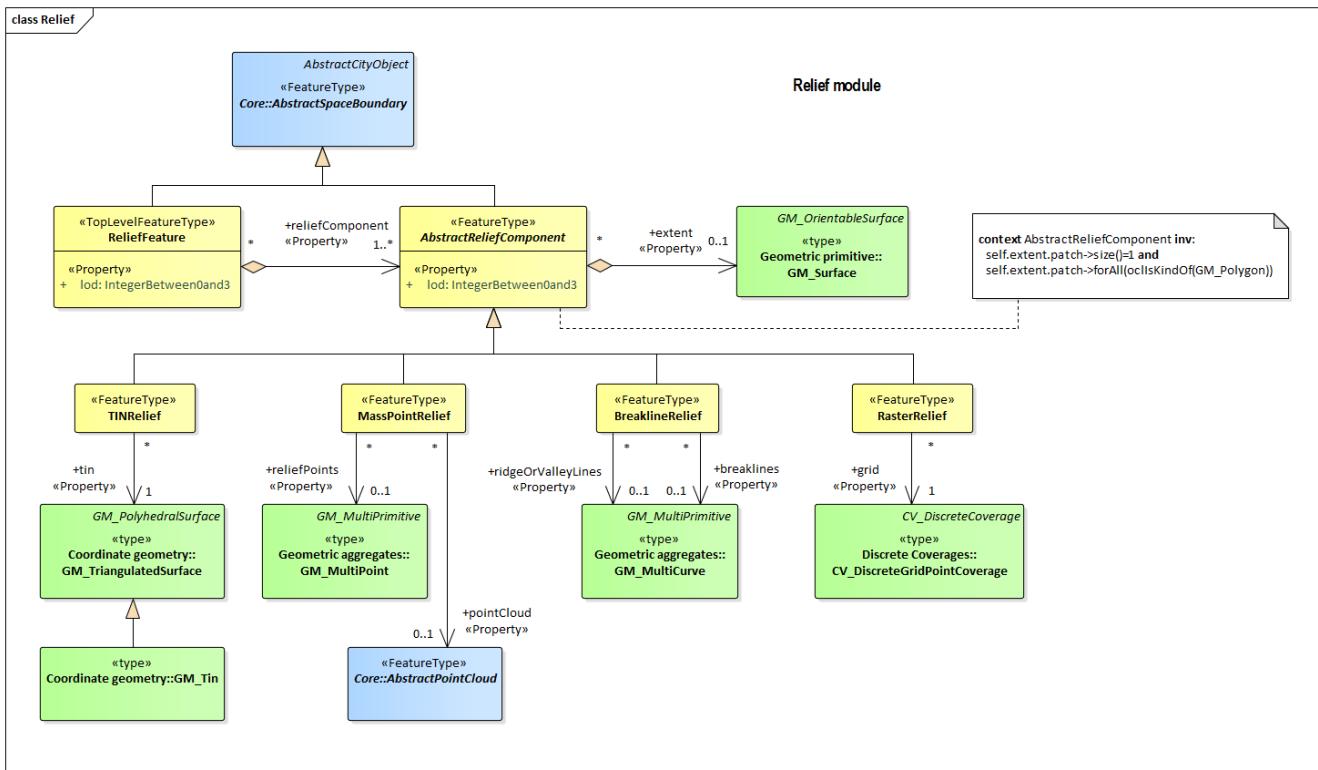


Figure 16. UML diagram of Relief module.

The [UML diagram of Relief module](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.5. Relief Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.5.1. Class AbstractReliefComponent

Subclass of [AbstractSpaceBoundary](#)

Requirement 59	/req/Relief/AbstractReliefComponent
----------------	-------------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractReliefComponent UML class as documented in the AbstractReliefComponent section of the Relief Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractReliefComponent UML class as documented in the AbstractReliefComponent section of the Relief Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractReliefComponent UML class; including the name, definition, type, and cardinality of those documented in the AbstractReliefComponent section of the Relief Data Dictionary .

Class AbstractReliefComponent

Definition:

Subtype Of: [AbstractSpaceBoundary](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractReliefComponent](#)

Target Role:

Target Class: [AbstractSpaceBoundary](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractReliefComponent](#)

Target Role: extent

Target Class: [GM_Surface](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BreaklineRelief](#)

Target Role:

Target Class: [AbstractReliefComponent](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TINRelief
Target Role:	
Target Class:	AbstractReliefComponent
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	MassPointRelief
Target Role:	
Target Class:	AbstractReliefComponent
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	ReliefFeature
Target Role:	reliefComponent
Target Class:	AbstractReliefComponent
Name:	
Type:	NoteLink
Direction:	Source → Destination
Source Role:	
Source Class:	Note
Target Role:	
Target Class:	AbstractReliefComponent
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	RasterRelief
Target Role:	
Target Class:	AbstractReliefComponent
Attributes	
Attribute Name:	lod
Value Type:	IntegerBetween0and3
Definition:	SIG3D: Number denoting the LOD of the relief component. The LOD concept for the relief is defined in chapter ...
Multiplicity:	
Stereotype:	«Property»

12.5.2. Class BreaklineRelief

Subclass of [AbstractReliefComponent](#)

Requirement 60	/req/Relief/BreaklineRelief
A	The Implementation Specification SHALL contain an element with the same definition as the BreaklineRelief UML class as documented in the BreaklineRelief section of the Relief Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BreaklineRelief UML class as documented in the BreaklineRelief section of the Relief Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BreaklineRelief UML class; including the name, definition, type, and cardinality of those documented in the BreaklineRelief section of the Relief Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BreaklineRelief UML class; including the name, definition, type, and cardinality of those documented in the BreaklineRelief section of the Relief Data Dictionary .

Class BreaklineRelief

Definition:

Subtype Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [BreaklineRelief](#)

Target Role: ridgeOrValleyLines

Target Class: [GM_MultiCurve](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BreaklineRelief](#)

Target Role:

Target Class: [AbstractReliefComponent](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	BreaklineRelief
Target Role:	breaklines
Target Class:	GM_MultiCurve
Attributes	

12.5.3. Class MassPointRelief

Subclass of [AbstractReliefComponent](#)

Requirement 61	/req/Relief/MassPointRelief
A	The Implementation Specification SHALL contain an element with the same definition as the MassPointRelief UML class as documented in the MassPointRelief section of the Relief Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the MassPointRelief UML class as documented in the MassPointRelief section of the Relief Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the MassPointRelief UML class; including the name, definition, type, and cardinality of those documented in the MassPointRelief section of the Relief Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the MassPointRelief UML class; including the name, definition, type, and cardinality of those documented in the MassPointRelief section of the Relief Data Dictionary .

Class MassPointRelief
Definition:
Subtype Of: AbstractReliefComponent
Stereotype: «FeatureType»

Associations

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	MassPointRelief
Target Role:	reliefPoints
Target Class:	GM_MultiPoint
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	MassPointRelief
Target Role:	pointCloud
Target Class:	AbstractPointCloud
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	MassPointRelief
Target Role:	
Target Class:	AbstractReliefComponent
Attributes	

12.5.4. Class RasterRelief

Subclass of [AbstractReliefComponent](#)

Requirement 62	/req/Relief/RasterRelief
A	The Implementation Specification SHALL contain an element with the same definition as the RasterRelief UML class as documented in the RasterRelief section of the Relief Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the RasterRelief UML class as documented in the RasterRelief section of the Relief Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the RasterRelief UML class; including the name, definition, type, and cardinality of those documented in the RasterRelief section of the Relief Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the RasterRelief UML class; including the name, definition, type, and cardinality of those documented in the RasterRelief section of the Relief Data Dictionary .

Class RasterRelief

Definition:

Subtype Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [RasterRelief](#)

Target Role:

Target Class: [AbstractReliefComponent](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [RasterRelief](#)

Target Role: grid

Target Class: [CV_DiscreteGridPointCoverage](#)

Attributes

12.5.5. Class ReliefFeature

Subclass of [AbstractSpaceBoundary](#)

Requirement 63	/req/Relief/ReliefFeature
A	The Implementation Specification SHALL contain an element with the same definition as the ReliefFeature UML class as documented in the ReliefFeature section of the Relief Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ReliefFeature UML class as documented in the ReliefFeature section of the Relief Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ReliefFeature UML class; including the name, definition, type, and cardinality of those documented in the ReliefFeature section of the Relief Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ReliefFeature UML class; including the name, definition, type, and cardinality of those documented in the ReliefFeature section of the Relief Data Dictionary .

Class ReliefFeature

Definition:
 Subtype Of: [AbstractSpaceBoundary](#)
 Stereotype: «TopLevelFeatureType»

Associations

Name:
 Type: Association
 Direction: Source → Destination
 Source Role:
 Source Class: [ReliefFeature](#)
 Target Role: reliefComponent
 Target Class: [AbstractReliefComponent](#)

Name:
 Type: Generalization
 Direction: Source → Destination
 Source Role:
 Source Class: [ReliefFeature](#)
 Target Role:
 Target Class: [AbstractSpaceBoundary](#)

Attributes

Attribute Name: lod
 Value Type: IntegerBetween0and3
 Definition: SIG3D: Number denoting the LOD of the relief feature. The LOD concept for the relief is defined in chapter ...
 Multiplicity:
 Stereotype: «Property»

12.5.6. Class TINRelief

Subclass of [AbstractReliefComponent](#)

Requirement 64	/req/Relief/TINRelief
A	The Implementation Specification SHALL contain an element with the same definition as the TINRelief UML class as documented in the TINRelief section of the Relief Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TINRelief UML class as documented in the TINRelief section of the Relief Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TINRelief UML class; including the name, definition, type, and cardinality of those documented in the TINRelief section of the Relief Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the TINRelief UML class; including the name, definition, type, and cardinality of those documented in the TINRelief section of the Relief Data Dictionary .
---	--

Class TINRelief

Definition:

Subtype Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TINRelief](#)

Target Role:

Target Class: [AbstractReliefComponent](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TINRelief](#)

Target Role: tin

Target Class: [GM_TriangulatedSurface](#)

Attributes

12.5.7. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.6. Building Model

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-building>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

The building model is one of the most detailed thematic concepts of CityGML. It allows for the

representation of thematic and spatial aspects of buildings and building parts in five levels of detail, LOD0 to LOD4. The building model of CityGML is defined by the thematic extension module Building (cf. chapter 7). Fig. 26 provides examples of 3D city and building models in LOD1 – 4.

The UML diagram of the building model is depicted in [Building UML Diagram](#). The Data Dictionary for the Building Model Package is provided in section [Building Model Data Dictionary](#).

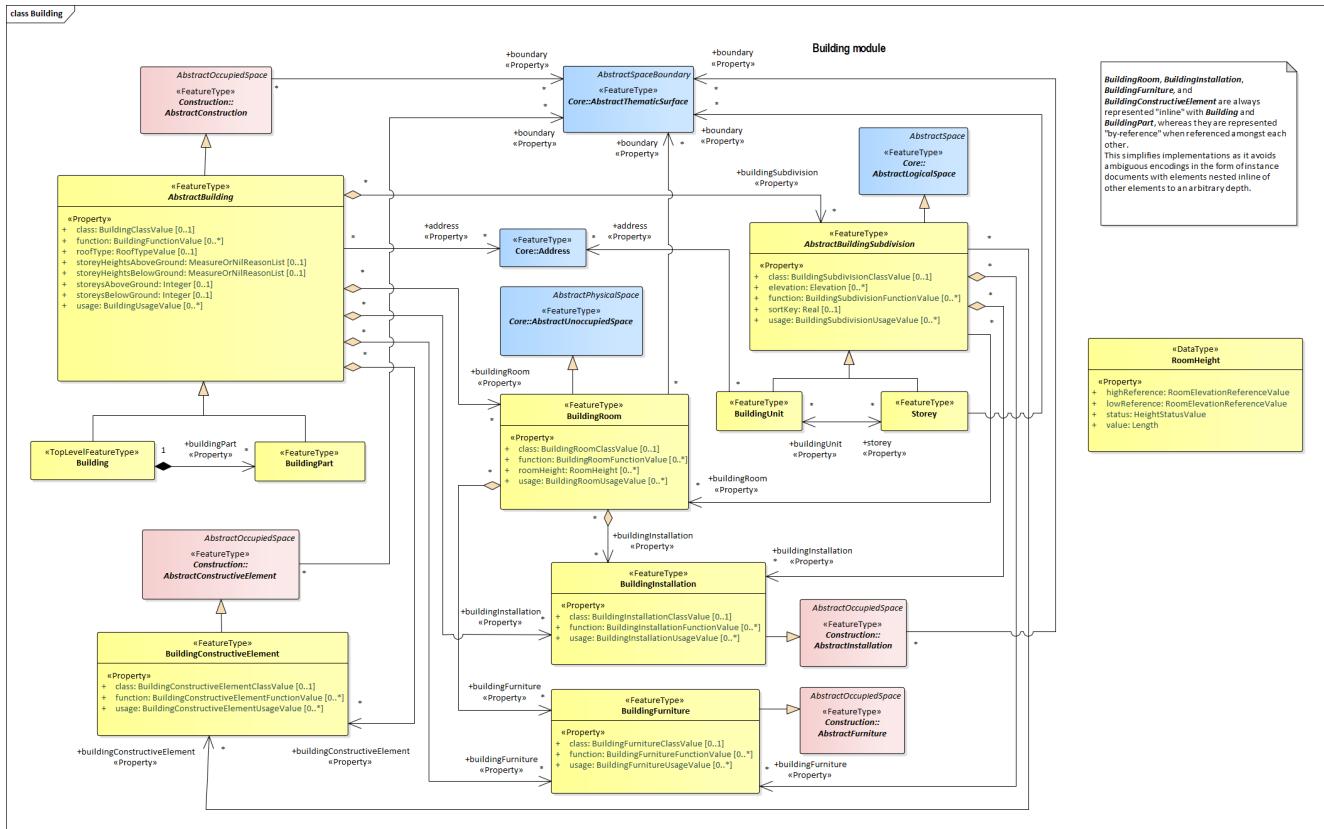


Figure 17. UML diagram of CityGML's building model.

The [UML diagram of CityGML's building model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.7. Building Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.7.1. Class AbstractBuilding

Subclass of [AbstractConstruction](#)

Requirement 65	/req/Building/AbstractBuilding
----------------	--------------------------------

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractBuilding UML class as documented in the AbstractBuilding section of the Building Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractBuilding UML class as documented in the AbstractBuilding section of the Building Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractBuilding UML class; including the name, definition, type, and cardinality of those documented in the AbstractBuilding section of the Building Data Dictionary .

Class AbstractBuilding

Definition:

Subtype Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBuilding](#)

Target Role: buildingConstructiveElement

Target Class: [BuildingConstructiveElement](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBuilding](#)

Target Role: address

Target Class: [Address](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBuilding](#)

Target Role: buildingSubdivision

Target Class: [AbstractBuildingSubdivision](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBuilding
Target Role:	
Target Class:	AbstractConstruction
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBuilding
Target Role:	buildingFurniture
Target Class:	BuildingFurniture
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBuilding
Target Role:	buildingRoom
Target Class:	BuildingRoom
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBuilding
Target Role:	buildingInstallation
Target Class:	BuildingInstallation
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	BuildingPart
Target Role:	
Target Class:	AbstractBuilding
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Building
Target Role:	
Target Class:	AbstractBuilding
Attributes	

Attribute Name:	class
Value Type:	BuildingClassValue
Definition:	SIG3D: Classification of Building or BuildingPart as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BuildingFunctionValue
Definition:	SIG3D: Specified function of Building or BuildingPart as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	roofType
Value Type:	RoofTypeValue
Definition:	bSI: Basic configuration of the roof in terms of the different roof shapes.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeyHeightsAboveGround
Value Type:	MeasureOrNilReasonList
Definition:	SIG3D: List of heights for each storey above ground.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeyHeightsBelowGround
Value Type:	MeasureOrNilReasonList
Definition:	SIG3D: List of heights for each storey below ground.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeysAboveGround
Value Type:	Integer
Definition:	SIG3D: Number of storeys mainly above ground.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeysBelowGround
Value Type:	Integer
Definition:	SIG3D: Number of storeys mainly below ground.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingUsageValue
Definition:	SIG3D: Actual usage of Building or BuildingPart as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.7.2. Class AbstractBuildingSubdivision

Subclass of [AbstractLogicalSpace](#)

Requirement 66	/req/Building/AbstractBuildingSubdivision
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractBuildingSubdivision UML class as documented in the AbstractBuildingSubdivision section of the Building Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractBuildingSubdivision UML class as documented in the AbstractBuildingSubdivision section of the Building Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractBuildingSubdivision UML class; including the name, definition, type, and cardinality of those documented in the AbstractBuildingSubdivision section of the Building Data Dictionary .

Class AbstractBuildingSubdivision

Definition:

Subtype Of: [AbstractLogicalSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractBuildingSubdivision](#)

Target Role:

Target Class: [AbstractLogicalSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBuildingSubdivision](#)

Target Role: buildingInstallation

Target Class: [BuildingInstallation](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuildingSubdivision</p> <p>Target Role: buildingFurniture</p> <p>Target Class: BuildingFurniture</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuildingSubdivision</p> <p>Target Role: buildingRoom</p> <p>Target Class: BuildingRoom</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuildingSubdivision</p> <p>Target Role: buildingConstructiveElement</p> <p>Target Class: BuildingConstructiveElement</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Storey</p> <p>Target Role:</p> <p>Target Class: AbstractBuildingSubdivision</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuilding</p> <p>Target Role: buildingSubdivision</p> <p>Target Class: AbstractBuildingSubdivision</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BuildingUnit</p> <p>Target Role:</p> <p>Target Class: AbstractBuildingSubdivision</p>
Attributes

Attribute Name:	class
Value Type:	BuildingSubdivisionClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	elevation
Value Type:	Elevation
Definition:	[INSPIRE] Vertically-constrained dimensional property consisting of an absolute measure referenced to a well-defined surface which is commonly taken as origin (geoid, water level, etc.).
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BuildingSubdivisionFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	sortKey
Value Type:	Real
Definition:	Defines an order among the building unit objects, e.g. for sorting storeys.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingSubdivisionUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.7.3. Class Building

Subclass of [AbstractBuilding](#)

Requirement 67	/req/Building/Building
A	The Implementation Specification SHALL contain an element with the same definition as the Building UML class as documented in the Building section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Building UML class as documented in the Building section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Building UML class; including the name, definition, type, and cardinality of those documented in the Building section of the Building Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the Building UML class; including the name, definition, type, and cardinality of those documented in the Building section of the Building Data Dictionary .
---	--

Class Building

Definition:

Subtype Of: [AbstractBuilding](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Building](#)

Target Role:

Target Class: [AbstractBuilding](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Building](#)

Target Role: buildingPart

Target Class: [BuildingPart](#)

Attributes

12.7.4. Class BuildingClassValue

Subclass of <– section,>>

Requirement 68	/req/Building/BuildingClassValue
A	Any use of the BuildingClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingClassValue UML class as documented in the BuildingClassValue section of the Building Data Dictionary .

Class BuildingClassValue

Definition:

Subtype Of: <– section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.5. Class BuildingConstructiveElement

Subclass of [AbstractConstructiveElement](#)

Requirement 69	/req/Building/BuildingConstructiveElement
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingConstructiveElement UML class as documented in the BuildingConstructiveElement section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingConstructiveElement UML class as documented in the BuildingConstructiveElement section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BuildingConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BuildingConstructiveElement section of the Building Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BuildingConstructiveElement section of the Building Data Dictionary .

Class BuildingConstructiveElement

Definition:

Subtype Of: [AbstractConstructiveElement](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BuildingConstructiveElement](#)

Target Role:

Target Class: [AbstractConstructiveElement](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBuilding](#)

Target Role: buildingConstructiveElement

Target Class: [BuildingConstructiveElement](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBuildingSubdivision
Target Role:	buildingConstructiveElement
Target Class:	BuildingConstructiveElement
Attributes	
Attribute Name:	class
Value Type:	BuildingConstructiveElementClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BuildingConstructiveElementFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingConstructiveElementUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.7.6. Class BuildingConstructiveElementClassValue

Subclass of <-- section,>>

Requirement 70	/req/Building/BuildingConstructiveElementClassValue
A	Any use of the BuildingConstructiveElementClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingConstructiveElementClassValue UML class as documented in the Building ConstructiveElementClassValue section of the Building Data Dictionary .

Class BuildingConstructiveElementClassValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»
Associations
Attributes

12.7.7. Class BuildingConstructiveElementFunctionValue

Subclass of <-- section,>>

Requirement 71	/req/Building/BuildingConstructiveElementFunctionValue
A	Any use of the BuildingConstructiveElementFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingConstructiveElementFunctionValue UML class as documented in the BuildingConstructiveElementFunctionValue section of the Building Data Dictionary .

Class BuildingConstructiveElementFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.8. Class BuildingConstructiveElementUsageValue

Subclass of <-- section,>>

Requirement 72	/req/Building/BuildingConstructiveElementUsageValue
A	Any use of the BuildingConstructiveElementUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingConstructiveElementUsageValue UML class as documented in the BuildingConstructiveElementUsageValue section of the Building Data Dictionary .

Class BuildingConstructiveElementUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.9. Class BuildingFunctionValue

Subclass of <-- section,>>

Requirement 73	/req/Building/BuildingFunctionValue
A	Any use of the BuildingFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFunctionValue UML class as documented in the BuildingFunctionValue section of the Building Data Dictionary .

Class BuildingFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.10. Class BuildingFurniture

Subclass of [AbstractFurniture](#)

Requirement 74	/req/Building/BuildingFurniture
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingFurniture UML class as documented in the BuildingFurniture section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingFurniture UML class as documented in the BuildingFurniture section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BuildingFurniture UML class; including the name, definition, type, and cardinality of those documented in the BuildingFurniture section of the Building Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingFurniture UML class; including the name, definition, type, and cardinality of those documented in the BuildingFurniture section of the Building Data Dictionary .

Class BuildingFurniture

Definition:

Subtype Of: [AbstractFurniture](#)

Stereotype: «FeatureType»

Associations

<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BuildingFurniture</p> <p>Target Role:</p> <p>Target Class: AbstractFurniture</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuildingSubdivision</p> <p>Target Role: buildingFurniture</p> <p>Target Class: BuildingFurniture</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BuildingRoom</p> <p>Target Role: buildingFurniture</p> <p>Target Class: BuildingFurniture</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuilding</p> <p>Target Role: buildingFurniture</p> <p>Target Class: BuildingFurniture</p>
<p>Attributes</p> <p>Attribute Name: class</p> <p>Value Type: BuildingFurnitureClassValue</p> <p>Definition: SIG3D: Classification of BuildingFurniture as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p> <p>Attribute Name: function</p> <p>Value Type: BuildingFurnitureFunctionValue</p> <p>Definition: SIG3D: Specified function of BuildingFurniture as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

Attribute Name:	usage
Value Type:	BuildingFurnitureUsageValue
Definition:	SIG3D: Actual usage of BuildingFurniture as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.7.11. Class BuildingFurnitureClassValue

Subclass of <-- section,>>

Requirement 75	/req/Building/BuildingFurnitureClassValue
A	Any use of the BuildingFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFurnitureClassValue UML class as documented in the BuildingFurnitureClassValue section of the Building Data Dictionary .

Class BuildingFurnitureClassValue

Definition:

Subtype Of: <-- section,>>

StereoType: «CodeList»

Associations

Attributes

12.7.12. Class BuildingFurnitureFunctionValue

Subclass of <-- section,>>

Requirement 76	/req/Building/BuildingFurnitureFunctionValue
A	Any use of the BuildingFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFurnitureFunctionValue UML class as documented in the BuildingFurnitureFunctionValue section of the Building Data Dictionary .

Class BuildingFurnitureFunctionValue

Definition:

Subtype Of: <-- section,>>

StereoType: «CodeList»

Associations

Attributes

12.7.13. Class BuildingFurnitureUsageValue

Subclass of <-- section,>>

Requirement 77	/req/Building/BuildingFurnitureUsageValue
A	Any use of the BuildingFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingFurnitureUsageValue UML class as documented in the BuildingFurnitureUsageValue section of the Building Data Dictionary .

Class BuildingFurnitureUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.14. Class BuildingInstallation

Subclass of [AbstractInstallation](#)

Requirement 78	/req/Building/BuildingInstallation
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingInstallation UML class as documented in the BuildingInstallation section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingInstallation UML class as documented in the BuildingInstallation section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BuildingInstallation UML class; including the name, definition, type, and cardinality of those documented in the BuildingInstallation section of the Building Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingInstallation UML class; including the name, definition, type, and cardinality of those documented in the BuildingInstallation section of the Building Data Dictionary .

Class BuildingInstallation

Definition:
Subtype Of: [AbstractInstallation](#)
Stereotype: «FeatureType»

Associations

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [BuildingInstallation](#)
Target Role:
Target Class: [AbstractInstallation](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractBuildingSubdivision](#)
Target Role: buildingInstallation
Target Class: [BuildingInstallation](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractBuilding](#)
Target Role: buildingInstallation
Target Class: [BuildingInstallation](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [BuildingRoom](#)
Target Role: buildingInstallation
Target Class: [BuildingInstallation](#)

Attributes

Attribute Name: class
Value Type: [BuildingInstallationClassValue](#)
Definition: SIG3D: Classification of BuildingInstallation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity: [0..1]
Stereotype: «Property»

Attribute Name:	function
Value Type:	BuildingInstallationFunctionValue
Definition:	SIG3D: Specified function of BuildingInstallation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]

Attribute Name:	usage
Value Type:	BuildingInstallationUsageValue
Definition:	SIG3D: Actual usage of BuildingInstallation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]

12.7.15. Class BuildingInstallationClassValue

Subclass of <-- section,>>

Requirement 79	/req/Building/BuildingInstallationClassValue
A	Any use of the BuildingInstallationClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingInstallationClassValue UML class as documented in the BuildingInstallationClassValue section of the Building Data Dictionary .

Class BuildingInstallationClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.16. Class BuildingInstallationFunctionValue

Subclass of <-- section,>>

Requirement 80	/req/Building/BuildingInstallationFunctionValue
A	Any use of the BuildingInstallationFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingInstallationFunctionValue UML class as documented in the BuildingInstallationFunctionValue section of the Building Data Dictionary .

Class BuildingInstallationFunctionValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.7.17. Class BuildingInstallationUsageValue

Subclass of <-- section,>>

Requirement 81	/req/Building/BuildingInstallationUsageValue
A	Any use of the BuildingInstallationUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingInstallationUsageValue UML class as documented in the BuildingInstallationUsageValue section of the Building Data Dictionary .

Class BuildingInstallationUsageValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.7.18. Class BuildingPart

Subclass of [AbstractBuilding](#)

Requirement 82	/req/Building/BuildingPart
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingPart UML class as documented in the BuildingPart section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingPart UML class as documented in the BuildingPart section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BuildingPart UML class; including the name, definition, type, and cardinality of those documented in the BuildingPart section of the Building Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingPart UML class; including the name, definition, type, and cardinality of those documented in the BuildingPart section of the Building Data Dictionary .
---	--

Class BuildingPart

Definition:

Subtype Of: [AbstractBuilding](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BuildingPart](#)

Target Role:

Target Class: [AbstractBuilding](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Building](#)

Target Role: buildingPart

Target Class: [BuildingPart](#)

Attributes

12.7.19. Class BuildingRoom

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 83	/req/Building/BuildingRoom
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingRoom UML class as documented in the BuildingRoom section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingRoom UML class as documented in the BuildingRoom section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BuildingRoom UML class; including the name, definition, type, and cardinality of those documented in the BuildingRoom section of the Building Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingRoom UML class; including the name, definition, type, and cardinality of those documented in the BuildingRoom section of the Building Data Dictionary .
---	--

Class BuildingRoom

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BuildingRoom](#)

Target Role:

Target Class: [AbstractUnoccupiedSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [BuildingRoom](#)

Target Role: buildingFurniture

Target Class: [BuildingFurniture](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [BuildingRoom](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [BuildingRoom](#)

Target Role: buildingInstallation

Target Class: [BuildingInstallation](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuildingSubdivision</p> <p>Target Role: buildingRoom</p> <p>Target Class: BuildingRoom</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuilding</p> <p>Target Role: buildingRoom</p> <p>Target Class: BuildingRoom</p>
Attributes
<p>Attribute Name: class</p> <p>Value Type: BuildingRoomClassValue</p> <p>Definition: SIG3D: Classification of Room as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: function</p> <p>Value Type: BuildingRoomFunctionValue</p> <p>Definition: SIG3D: Function of Room as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: roomHeight</p> <p>Value Type: RoomHeight</p> <p>Definition: Height of the room.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: usage</p> <p>Value Type: BuildingRoomUsageValue</p> <p>Definition: SIG3D: Usage of Room as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

12.7.20. Class BuildingRoomClassValue

Subclass of <← section,>>

Requirement 84	/req/Building/BuildingRoomClassValue
----------------	--------------------------------------

A	Any use of the BuildingRoomClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingRoomClassValue UML class as documented in the BuildingRoomClassValue section of the Building Data Dictionary .
---	---

Class BuildingRoomClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.21. Class BuildingRoomFunctionValue

Subclass of <-- section,>>

Requirement 85	/req/Building/BuildingRoomFunctionValue
A	Any use of the BuildingRoomFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingRoomFunctionValue UML class as documented in the BuildingRoomFunctionValue section of the Building Data Dictionary .

Class BuildingRoomFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.22. Class BuildingRoomUsageValue

Subclass of <-- section,>>

Requirement 86	/req/Building/BuildingRoomUsageValue
A	Any use of the BuildingRoomUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingRoomUsageValue UML class as documented in the BuildingRoomUsageValue section of the Building Data Dictionary .

Class BuildingRoomUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.23. Class BuildingSubdivisionClassValue

Subclass of <-- section,>>

Requirement 87	/req/Building/BuildingSubdivisionClassValue
A	Any use of the BuildingSubdivisionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingSubdivisionClassValue UML class as documented in the BuildingSubdivisionClassValue section of the Building Data Dictionary .

Class BuildingSubdivisionClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.24. Class BuildingSubdivisionFunctionValue

Subclass of <-- section,>>

Requirement 88	/req/Building/BuildingSubdivisionFunctionValue
A	Any use of the BuildingSubdivisionFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingSubdivisionFunctionValue UML class as documented in the BuildingSubdivisionFunctionValue section of the Building Data Dictionary .

Class BuildingSubdivisionFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.25. Class BuildingSubdivisionUsageValue

Subclass of <-- section,>>

Requirement 89	/req/Building/BuildingSubdivisionUsageValue
A	Any use of the BuildingSubdivisionUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingSubdivisionUsageValue UML class as documented in the BuildingSubdivisionUsageValue section of the Building Data Dictionary .

Class BuildingSubdivisionUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.26. Class BuildingUnit

Subclass of [AbstractBuildingSubdivision](#)

Requirement 90	/req/Building/BuildingUnit
A	The Implementation Specification SHALL contain an element with the same definition as the BuildingUnit UML class as documented in the BuildingUnit section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BuildingUnit UML class as documented in the BuildingUnit section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BuildingUnit UML class; including the name, definition, type, and cardinality of those documented in the BuildingUnit section of the Building Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BuildingUnit UML class; including the name, definition, type, and cardinality of those documented in the BuildingUnit section of the Building Data Dictionary .

Class BuildingUnit

Definition:
 Subtype Of: [AbstractBuildingSubdivision](#)
 Stereotype: «FeatureType»

Associations

Name:
 Type: Association
 Direction: Source → Destination
 Source Role:
 Source Class: [BuildingUnit](#)
 Target Role: address
 Target Class: [Address](#)

Name:
 Type: Generalization
 Direction: Source → Destination
 Source Role:
 Source Class: [BuildingUnit](#)
 Target Role:
 Target Class: [AbstractBuildingSubdivision](#)

Name:
 Type: Association
 Direction: Bi-Directional
 Source Role: storey
 Source Class: [Storey](#)
 Target Role: buildingUnit
 Target Class: [BuildingUnit](#)

Attributes

12.7.27. Class BuildingUsageValue

Subclass of <-- section,>>

Requirement 91	/req/Building/BuildingUsageValue
A	Any use of the BuildingUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BuildingUsageValue UML class as documented in the BuildingUsageValue section of the Building Data Dictionary .

Class BuildingUsageValue

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «CodeList»

Associations

Attributes

12.7.28. Class RoofTypeValue

Subclass of <-- section,>>

Requirement 92	/req/Building/RoofTypeValue
A	Any use of the RoofTypeValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoofTypeValue UML class as documented in the RoofTypeValue section of the Building Data Dictionary .

Class RoofTypeValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.29. Class RoomElevationReferenceValue

Subclass of <-- section,>>

Requirement 93	/req/Building/RoomElevationReferenceValue
A	Any use of the RoomElevationReferenceValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoomElevationReferenceValue UML class as documented in the RoomElevationReferenceValue section of the Building Data Dictionary .

Class RoomElevationReferenceValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.7.30. Class Storey

Subclass of [AbstractBuildingSubdivision](#)

Requirement 94	/req/Building/Storey
A	The Implementation Specification SHALL contain an element with the same definition as the Storey UML class as documented in the Storey section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Storey UML class as documented in the Storey section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Storey UML class; including the name, definition, type, and cardinality of those documented in the Storey section of the Building Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Storey UML class; including the name, definition, type, and cardinality of those documented in the Storey section of the Building Data Dictionary .

Class Storey

Definition:

Subtype Of: [AbstractBuildingSubdivision](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Bi-Directional

Source Role: storey

Source Class: [Storey](#)

Target Role: buildingUnit

Target Class: [BuildingUnit](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Storey](#)

Target Role:

Target Class: [AbstractBuildingSubdivision](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Storey](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

Attributes

12.7.31. Class RoomHeight

Subclass of <-- section,>>

Requirement 95	/req/Building/RoomHeight
A	Any use of the RoomHeight type in the Implementation Specification SHALL have the same definition as the RoomHeight UML class as documented in the RoomHeight section of the Building Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the RoomHeight UML class as documented in the RoomHeight section of the Building Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the RoomHeight UML class; including the name, definition, type, and cardinality of those documented in the RoomHeight section of the Building Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the RoomHeight UML class; including the name, definition, type, and cardinality of those documented in the RoomHeight section of the Building Data Dictionary .

Class RoomHeight

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Attributes

Attribute Name: highReference

Value Type: RoomElevationReferenceValue

Definition: [INSPIRE] Element used as the high reference.

Multiplicity:

Stereotype: «Property»

Attribute Name: lowReference

Value Type: RoomElevationReferenceValue

Definition: [INSPIRE] Element as the low reference.

Multiplicity:

Stereotype: «Property»

Attribute Name:	status
Value Type:	HeightStatusValue
Definition:	[INSPIRE] The way the height has been captured.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	value
Value Type:	Length
Definition:	Value of the room height.
Multiplicity:	
Stereotype:	«Property»

12.7.32. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

NOTE The following content is from the previous draft. Is this sort of content useful?

BuildingType, Building

The Building class is one of the two subclasses of _AbstractBuilding. If a building only consists of one (homogeneous) part, this class shall be used. A building composed of structural segments differing in, for example the number of storeys or the roof type has to be separated into one Building having one or more additional BuildingPart (see Fig. 28). The geometry and non-spatial properties of the central part of the building should be represented in the aggregating Building feature.

BuildingType, Building Part

The class BuildingPart is derived from _AbstractBuilding. It is used to model a structural part of a building (see Fig. 28). A BuildingPart object should be uniquely related to exactly one building or building part object.

AbstractBuildingType, _AbstractBuilding

The abstract class _AbstractBuilding contains properties for building attributes, purely geometric representations, and geometric/semantic representations of the building or building part in different levels of detail. The attributes describe:

1. classification of the building or building part (class), the different intended usages (function), and the different actual usages (usage). The permitted values for these attributes can be specified in code lists.
2. The year of construction (yearOfConstruction) and the year of demolition (yearOfDemolition) of the building or building part. These attributes can be used to describe the chronology of the

building development within a city model. The points of time refer to real world time.

3. The roof type of the building or building part (roofType). The permitted values for this attribute can be specified in a code list.
4. The measured relative height (measuredHeight) of the building or building part.
5. The number of storeys above (storeyAboveGround) and below (storeyBelowGround) ground level.
6. The list of storey heights above (storeyHeightsAboveGround) and below (storeyHeightsBelowGround) ground level. The first value in a list denotes the height of the nearest storey wrt. to the ground level and last value the height of the farthest.

Table 6. Semantic themes of the class _AbstractBuilding

Geometric / semantic theme	Property type	LOD0	LOD1	LOD2	LOD3	LOD4
Building footprint and roof edge	gml:MultiSurfaceType	•				
Volume part of the building shell	gml:SolidType		•	•	•	•
Surface part of the building shell	gml:MultiSurfaceType		•	•	•	•
Terrain intersection curve	gml:MultiCurveType		•	•	•	•
Curve part of the building shell	gml:MultiCurveType			•	•	•
Building parts	BuildingPartType		•	•	•	•
Boundary surfaces (chapter 10.3.3)	AbstractBoundarySurfaceType			•	•	•
Outer building installations (chapter 10.3.2)	BuildingInstallationType			•	•	•
Openings (chapter 10.3.4)	AbstractOpeningType				•	•
Rooms (chapter 10.3.5)	RoomType					•
Interior building installations (chapter 10.3.5)	IntBuildingInstallationType					•

BuildingInstallationType, BuildingInstallation

Note: insert BuildingInstallation UML

12.7.33. Boundary surfaces

AbstractBoundarySurfaceType, _BoundarySurface

NOTE Insert AbstractBoundarySurfaceType, _BoundarySurface UML

GroundSurfaceType, GroundSurface

NOTE insert GroundSurfaceType, GroundSurface uml

The ground plate of a building or building part is modelled by the class GroundSurface. The polygon defining the ground plate is congruent with the building's footprint. However, the surface normal of the ground plate is pointing downwards.

OuterCeilingSurfaceType, OuterCeilingSurface

NOTE insert OuterCeilingSurfaceType, OuterCeilingSurface UML

A mostly horizontal surface belonging to the outer building shell and having the orientation pointing downwards can be modeled as an OuterCeilingSurface. Examples are the visible part of the ceiling of a loggia or the ceiling of a passage.

WallSurfaceType, WallSurface

NOTE insert WallSurfaceType, WallSurface UML

All parts of the building facade belonging to the outer building shell can be modelled by the class WallSurface.

OuterFloorSurfaceType, OuterFloorSurface

NOTE insert OuterFloorSurfaceType, OuterFloorSurface UML

A mostly horizontal surface belonging to the outer building shell and with the orientation pointing upwards can be modeled as an OuterFloorSurface. An example is the floor of a loggia.

RoofSurfaceType, RoofSurface

NOTE insert RoofSurfaceType, RoofSurface UML

The major roof parts of a building or building part are expressed by the class RoofSurface. Secondary parts of a roof with a specific semantic meaning like dormers or chimneys should be modelled as BuildingInstallation.

ClosureSurfaceType, ClosureSurface

NOTE insert ClosureSurfaceType, ClosureSurface UML

An opening in a building not filled by a door or window can be sealed by a virtual surface called ClosureSurface (cf. chapter 6.4). Hence, buildings with open sides like a barn or a hangar, can be virtually closed in order to be able to compute their volume. ClosureSurfaces are also used in the interior building model. If two rooms with a different function (e.g. kitchen and living room) are directly connected without a separating door, a ClosureSurface should be used to separate or connect the volumes of both rooms.

FloorSurfaceType, FloorSurface

NOTE insert FloorSurfaceType, FloorSurface UML

The class FloorSurface must only be used in the LOD4 interior building model for modelling the floor of a room.

InteriorWallSurfaceType, InteriorWallSurface

NOTE insert InteriorWallSurfaceType, InteriorWallSurface UML

The class InteriorWallSurface must only be used in the LOD4 interior building model for modelling the visible surfaces of the room walls.

CeilingSurfaceType, CeilingSurface

NOTE Insert CeilingSurfaceType, CeilingSurface UML

The class CeilingSurface must only be used in the LOD4 interior building model for modelling the ceiling of a room.

12.7.34. Openings

AbstractOpeningType, _Opening

NOTE insert AbstractOpeningType, _Opening UML

The class _Opening is the abstract base class for semantically describing openings like doors or windows in outer or inner boundary surfaces like walls and roofs. Openings only exist in models of LOD3 or LOD4. Each _Opening is associated with a *gml:MultiSurface* geometry. Alternatively, the geometry may be given as *ImplicitGeometry* object. Following the concept of *ImplicitGeometry* the geometry of a prototype opening is stored only once in a local coordinate system and referenced by other opening features (see chapter 8.2).

WindowType, Window

NOTE insert WindowType, Window UML

The class Window is used for modelling windows in the exterior shell of a building, or hatches between adjacent rooms. The formal difference between the classes Window and Door is that – in normal cases – Windows are not specifically intended for the transit of people or vehicles.

DoorType, Door

NOTE insert DoorType, Door UML

The class Door is used for modelling doors in the exterior shell of a building, or between adjacent rooms. Doors can be used by people to enter or leave a building or room. In contrast to a ClosureSurface a door may be closed, blocking the transit of people. A Door may be assigned zero or more addresses. The corresponding Address-PropertyType is defined within the CityGML core module (cf. chapter 10.1.4).

12.7.35. Building Interior

RoomType, Room

NOTE insert RoomType, Room UML

A Room is a semantic object for modelling the free space inside a building and should be uniquely related to exactly one building or building part object. It should be closed (if necessary by using ClosureSurfaces) and the geometry normally will be described by a solid (lod4Solid). However, if the topological correctness of the boundary cannot be guaranteed, the geometry can alternatively be given as a MultiSurface (lod4MultiSurface). The surface normals of the outer shell of a GML solid must point outwards. This is important to consider when Room surfaces should be assigned Appearances. In this case, textures and colors must be placed on the backside of the corresponding surfaces in order to be visible from the inside of the room.

BuildingFurnitureType, BuildingFurniture

NOTE insert BuildingFurnitureType, BuildingFurniture UML

Rooms may have BuildingFurnitures and IntBuildingInstallations. A BuildingFurniture is a movable part of a room, such as a chair or furniture. A BuildingFurniture object should be uniquely related to exactly one room object. Its geometry may be represented by an explicit geometry or an ImplicitGeometry object. Following the concept of ImplicitGeometry the geometry of a prototype building furniture is stored only once in a local coordinate system and referenced by other building furniture features (see chapter 8.2).

IntBuildingInstallationType, IntBuildingInstallation

NOTE insert IntBuildingInstallationType, IntBuildingInstallation UML

12.7.36. Modelling building storeys using CityObjectGroups

CityGML does currently not provide a specific concept for the representation of storeys as it is available in the AEC/FM standard IFC (IAI 2006). However, a storey can be represented as an explicit aggregation of all building features on a certain height level using CityGML's notion of CityObjectGroups (cf. chapter 10.11).

In order to model building storeys with CityGML's generic grouping concept, a nested hierarchy of

CityObject-Group objects has to be used. In a first step, all semantic objects belonging to a specific storey are grouped. The attributes of the corresponding CityObjectGroup object are set as follows:

- The class attribute shall be assigned the value “building separation”.
- The function attribute shall be assigned the value “lodXStorey” with X between 1 and 4 in order to de-note that this group represents a storey wrt. a specific LOD.
- The storey name or number can be stored in the gml:name property. The storey number attribute shall be assigned the value “storeyNo_X” with decimal number X in order to denote that this group represents a storey wrt. a specific number.

In a second step, the CityObjectGroup objects representing different storeys are grouped themselves. By using the generic aggregation concept of CityObjectGroup, the “storeys group” is associated with the corresponding Building or BuildingPart object. The class attribute of the storeys group shall be assigned the value “building storeys”.

12.8. Tunnel

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-tunnel	
Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Tunnel Model is depicted in [Tunnel UML Diagram](#). The Data Dictionary for the Tunnel Package is provided in section [Tunnel Model Data Dictionary](#).

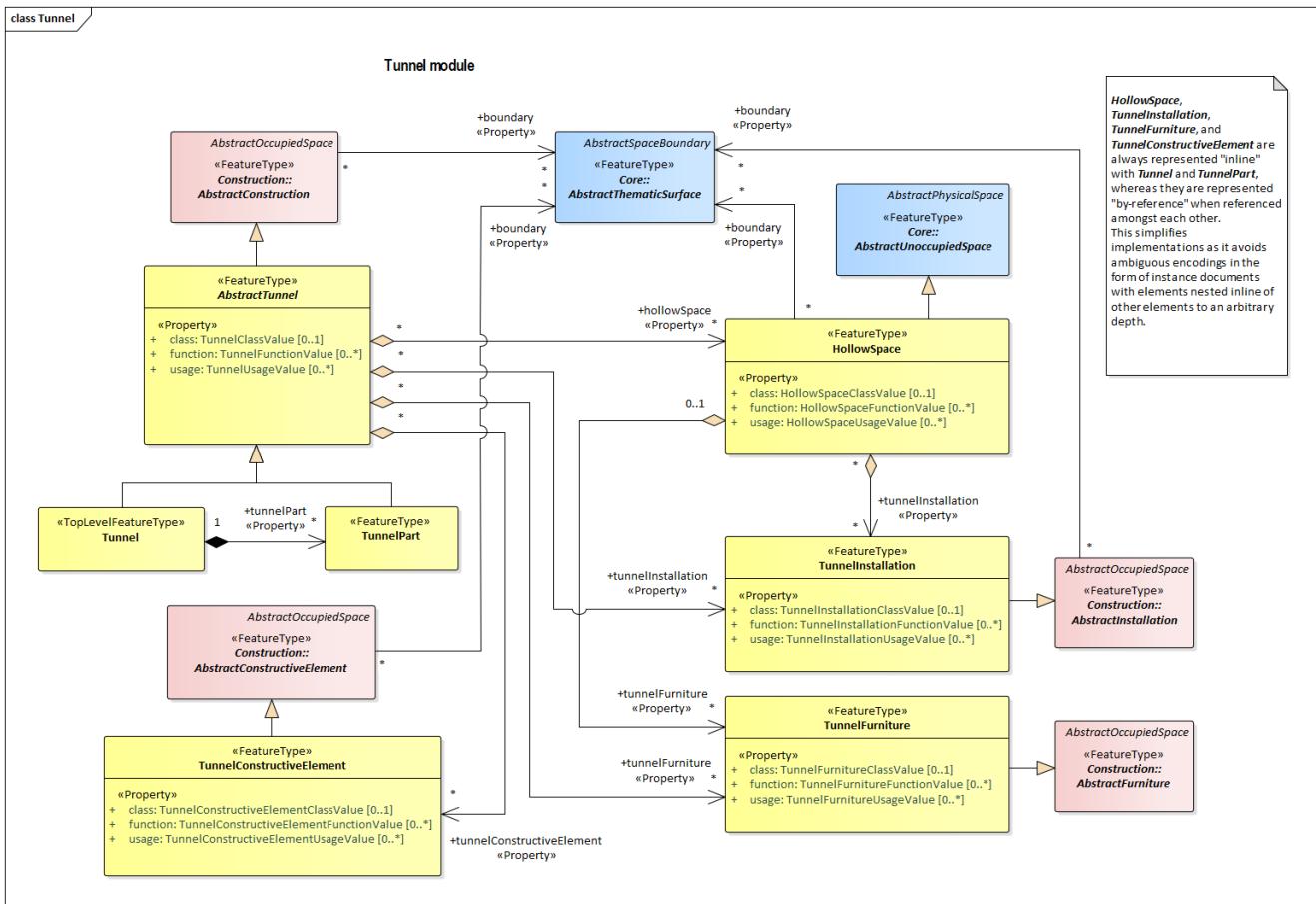


Figure 18. UML diagram of the Tunnel Model.

The [UML diagram of the Tunnel Model](#). is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.9. Tunnel Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.9.1. Class AbstractTunnel

Subclass of [AbstractConstruction](#)

Requirement 96	/req/Tunnel/AbstractTunnel
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTunnel UML class as documented in the AbstractTunnel section of the Tunnel Data Dictionary .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTunnel UML class as documented in the AbstractTunnel section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTunnel UML class; including the name, definition, type, and cardinality of those documented in the AbstractTunnel section of the Tunnel Data Dictionary .

Class AbstractTunnel

Definition:

Subtype Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role: tunnelFurniture

Target Class: [TunnelFurniture](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role:

Target Class: [AbstractConstruction](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role: tunnelInstallation

Target Class: [TunnelInstallation](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role: tunnelConstructiveElement

Target Class: [TunnelConstructiveElement](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractTunnel</p> <p>Target Role: hollowSpace</p> <p>Target Class: HollowSpace</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: TunnelPart</p> <p>Target Role:</p> <p>Target Class: AbstractTunnel</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Tunnel</p> <p>Target Role:</p> <p>Target Class: AbstractTunnel</p>
<p>Attributes</p> <p>Attribute Name: class</p> <p>Value Type: TunnelClassValue</p> <p>Definition: SIG3D: Classification of the actual usage of Tunnel or TunnelPart as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: function</p> <p>Value Type: TunnelFunctionValue</p> <p>Definition: SIG3D: Specified function of Tunnel or TunnelPart as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: usage</p> <p>Value Type: TunnelUsageValue</p> <p>Definition: SIG3D: Actual usage of Tunnel or TunnelPart as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

12.9.2. Class HollowSpace

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 97	/req/Tunnel/HollowSpace
A	The Implementation Specification SHALL contain an element with the same definition as the HollowSpace UML class as documented in the HollowSpace section of the Tunnel Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the HollowSpace UML class as documented in the HollowSpace section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the HollowSpace UML class; including the name, definition, type, and cardinality of those documented in the HollowSpace section of the Tunnel Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the HollowSpace UML class; including the name, definition, type, and cardinality of those documented in the HollowSpace section of the Tunnel Data Dictionary .

Class HollowSpace

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [HollowSpace](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [HollowSpace](#)

Target Role: tunnelFurniture

Target Class: [TunnelFurniture](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [HollowSpace](#)

Target Role:

Target Class: [AbstractUnoccupiedSpace](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	HollowSpace
Target Role:	tunnelInstallation
Target Class:	TunnelInstallation
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractTunnel
Target Role:	hollowSpace
Target Class:	HollowSpace
Attributes	
Attribute Name:	class
Value Type:	HollowSpaceClassValue
Definition:	SIG3D: Classification of HollowSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	HollowSpaceFunctionValue
Definition:	SIG3D: Specified function of HollowSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	HollowSpaceUsageValue
Definition:	SIG3D: Actual usage of HollowSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.9.3. Class HollowSpaceClassValue

Subclass of <– section,>>

Requirement 98	/req/Tunnel/HollowSpaceClassValue
A	Any use of the HollowSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HollowSpaceClassValue UML class as documented in the HollowSpaceClassValue section of the Tunnel Data Dictionary .

Class HollowSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.4. Class HollowSpaceFunctionValue

Subclass of <-- section,>>

Requirement 99	/req/Tunnel/HollowSpaceFunctionValue
A	Any use of the HollowSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HollowSpaceFunctionValue UML class as documented in the HollowSpaceFunctionValue section of the Tunnel Data Dictionary .

Class HollowSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.5. Class HollowSpaceUsageValue

Subclass of <-- section,>>

Requirement 100	/req/Tunnel/HollowSpaceUsageValue
A	Any use of the HollowSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HollowSpaceUsageValue UML class as documented in the HollowSpaceUsageValue section of the Tunnel Data Dictionary .

Class HollowSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.6. Class Tunnel

Subclass of [AbstractTunnel](#)

Requirement 101	/req/Tunnel/Tunnel
A	The Implementation Specification SHALL contain an element with the same definition as the Tunnel UML class as documented in the Tunnel section of the Tunnel Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Tunnel UML class as documented in the Tunnel section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Tunnel UML class; including the name, definition, type, and cardinality of those documented in the Tunnel section of the Tunnel Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Tunnel UML class; including the name, definition, type, and cardinality of those documented in the Tunnel section of the Tunnel Data Dictionary .

Class Tunnel

Definition:

Subtype Of: [AbstractTunnel](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Tunnel](#)

Target Role:

Target Class: [AbstractTunnel](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Tunnel](#)

Target Role: tunnelPart

Target Class: [TunnelPart](#)

Attributes

12.9.7. Class TunnelClassValue

Subclass of <-- section,>>

Requirement 102	/req/Tunnel/TunnelClassValue
A	Any use of the TunnelClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelClassValue UML class as documented in the TunnelClassValue section of the Tunnel Data Dictionary .

Class TunnelClassValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»
Associations
Attributes

12.9.8. Class TunnelConstructiveElement

Subclass of [AbstractConstructiveElement](#)

Requirement 103	/req/Tunnel/TunnelConstructiveElement
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelConstructiveElement UML class as documented in the TunnelConstructiveElement section of the Tunnel Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelConstructiveElement UML class as documented in the TunnelConstructiveElement section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TunnelConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the TunnelConstructiveElement section of the Tunnel Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the TunnelConstructiveElement section of the Tunnel Data Dictionary .

Class TunnelConstructiveElement

Definition:

Subtype Of: [AbstractConstructiveElement](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TunnelConstructiveElement](#)

Target Role:

Target Class: [AbstractConstructiveElement](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role: tunnelConstructiveElement

Target Class: [TunnelConstructiveElement](#)

Attributes

Attribute Name: class

Value Type: [TunnelConstructiveElementClassValue](#)

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: [TunnelConstructiveElementFunctionValue](#)

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

Attribute Name: usage

Value Type: [TunnelConstructiveElementUsageValue](#)

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

12.9.9. Class TunnelConstructiveElementClassValue

Subclass of <– section,>>

Requirement 104	/req/Tunnel/TunnelConstructiveElementClassValue
-----------------	---

A	Any use of the TunnelConstructiveElementClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelConstructiveElementClassValue UML class as documented in the TunnelConstructiveElementClassValue section of the Tunnel Data Dictionary .
---	--

Class TunnelConstructiveElementClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.10. Class TunnelConstructiveElementFunctionValue

Subclass of <-- section,>>

Requirement 105	/req/Tunnel/TunnelConstructiveElementFunctionValue
A	Any use of the TunnelConstructiveElementFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelConstructiveElementFunctionValue UML class as documented in the TunnelConstructiveElementFunctionValue section of the Tunnel Data Dictionary .

Class TunnelConstructiveElementFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.11. Class TunnelConstructiveElementUsageValue

Subclass of <-- section,>>

Requirement 106	/req/Tunnel/TunnelConstructiveElementUsageValue
-----------------	---

A	Any use of the TunnelConstructiveElementUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelConstructiveElementUsageValue UML class as documented in the TunnelConstructiveElementUsageValue section of the Tunnel Data Dictionary .
---	--

Class TunnelConstructiveElementUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.12. Class TunnelFunctionValue

Subclass of <-- section,>>

Requirement 107	/req/Tunnel/TunnelFunctionValue
A	Any use of the TunnelFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFunctionValue UML class as documented in the TunnelFunctionValue section of the Tunnel Data Dictionary .

Class TunnelFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.13. Class TunnelFurniture

Subclass of [AbstractFurniture](#)

Requirement 108	/req/Tunnel/TunnelFurniture
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelFurniture UML class as documented in the TunnelFurniture section of the Tunnel Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelFurniture UML class as documented in the TunnelFurniture section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TunnelFurniture UML class; including the name, definition, type, and cardinality of those documented in the TunnelFurniture section of the Tunnel Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelFurniture UML class; including the name, definition, type, and cardinality of those documented in the TunnelFurniture section of the Tunnel Data Dictionary .

Class TunnelFurniture

Definition:

Subtype Of: [AbstractFurniture](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TunnelFurniture](#)

Target Role:

Target Class: [AbstractFurniture](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role: tunnelFurniture

Target Class: [TunnelFurniture](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [HollowSpace](#)

Target Role: tunnelFurniture

Target Class: [TunnelFurniture](#)

Attributes

Attribute Name: class
 Value Type: TunnelFurnitureClassValue
 Definition: SIG3D: Classification of TunnelFurniture as given by the relevant national regulations, information communities, or specific applications.
 Multiplicity: [0..1]
 Stereotype: «Property»

Attribute Name: function
 Value Type: TunnelFurnitureFunctionValue
 Definition: SIG3D: Specified function of TunnelFurniture as given by the relevant national regulations, information communities, or specific applications.
 Multiplicity: [0..*]
 Stereotype: «Property»

Attribute Name: usage
 Value Type: TunnelFurnitureUsageValue
 Definition: SIG3D: Actual usage of TunnelFurniture as given by the relevant national regulations, information communities, or specific applications.
 Multiplicity: [0..*]
 Stereotype: «Property»

12.9.14. Class TunnelFurnitureClassValue

Subclass of <-- section,>>

Requirement 109	/req/Tunnel/TunnelFurnitureClassValue
A	Any use of the TunnelFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFurnitureClassValue UML class as documented in the TunnelFurnitureClassValue section of the Tunnel Data Dictionary .

Class TunnelFurnitureClassValue

Definition:
 Subtype Of: <-- section,>>
 StereoType: «CodeList»

Associations

Attributes

12.9.15. Class TunnelFurnitureFunctionValue

Subclass of <-- section,>>

Requirement 110	/req/Tunnel/TunnelFurnitureFunctionValue
-----------------	--

A	Any use of the TunnelFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFurnitureFunctionValue UML class as documented in the TunnelFurnitureFunctionValue section of the Tunnel Data Dictionary .
---	---

Class TunnelFurnitureFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.16. Class TunnelFurnitureUsageValue

Subclass of <-- section,>>

Requirement 111	/req/Tunnel/TunnelFurnitureUsageValue
A	Any use of the TunnelFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelFurnitureUsageValue UML class as documented in the TunnelFurnitureUsageValue section of the Tunnel Data Dictionary .

Class TunnelFurnitureUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.17. Class TunnelInstallation

Subclass of [AbstractInstallation](#)

Requirement 112	/req/Tunnel/TunnelInstallation
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelInstallation UML class as documented in the TunnelInstallation section of the Tunnel Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelInstallation UML class as documented in the TunnelInstallation section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TunnelInstallation UML class; including the name, definition, type, and cardinality of those documented in the TunnelInstallation section of the Tunnel Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelInstallation UML class; including the name, definition, type, and cardinality of those documented in the TunnelInstallation section of the Tunnel Data Dictionary .

Class TunnelInstallation

Definition:

Subtype Of: [AbstractInstallation](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TunnelInstallation](#)

Target Role:

Target Class: [AbstractInstallation](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTunnel](#)

Target Role: tunnelInstallation

Target Class: [TunnelInstallation](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [HollowSpace](#)

Target Role: tunnelInstallation

Target Class: [TunnelInstallation](#)

Attributes

Attribute Name: class
 Value Type: TunnelInstallationClassValue
 Definition: SIG3D: Classification of TunnelInstallation as given by the relevant national regulations, information communities, or specific applications.
 Multiplicity: [0..1]
 Stereotype: «Property»

Attribute Name: function
 Value Type: TunnelInstallationFunctionValue
 Definition: SIG3D: Specified function of TunnelInstallation as given by the relevant national regulations, information communities, or specific applications.
 Multiplicity: [0..*]
 Stereotype: «Property»

Attribute Name: usage
 Value Type: TunnelInstallationUsageValue
 Definition: SIG3D: Actual usage of TunnelInstallation as given by the relevant national regulations, information communities, or specific applications.
 Multiplicity: [0..*]
 Stereotype: «Property»

12.9.18. Class TunnelInstallationClassValue

Subclass of <-- section,>>

Requirement 113	/req/Tunnel/TunnelInstallationClassValue
A	Any use of the TunnelInstallationClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelInstallationClassValue UML class as documented in the TunnelInstallationClassValue section of the Tunnel Data Dictionary .

Class TunnelInstallationClassValue

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «CodeList»

Associations

Attributes

12.9.19. Class TunnelInstallationFunctionValue

Subclass of <-- section,>>

Requirement 114	/req/Tunnel/TunnelInstallationFunctionValue
-----------------	---

A	Any use of the TunnelInstallationFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelInstallationFunctionValue UML class as documented in the TunnelInstallationFunctionValue section of the Tunnel Data Dictionary .
---	--

Class TunnelInstallationFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.20. Class TunnelInstallationUsageValue

Subclass of <-- section,>>

Requirement 115	/req/Tunnel/TunnelInstallationUsageValue
A	Any use of the TunnelInstallationUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelInstallationUsageValue UML class as documented in the TunnelInstallationUsageValue section of the Tunnel Data Dictionary .

Class TunnelInstallationUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.21. Class TunnelPart

Subclass of [AbstractTunnel](#)

Requirement 116	/req/Tunnel/TunnelPart
A	The Implementation Specification SHALL contain an element with the same definition as the TunnelPart UML class as documented in the TunnelPart section of the Tunnel Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TunnelPart UML class as documented in the TunnelPart section of the Tunnel Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TunnelPart UML class; including the name, definition, type, and cardinality of those documented in the TunnelPart section of the Tunnel Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TunnelPart UML class; including the name, definition, type, and cardinality of those documented in the TunnelPart section of the Tunnel Data Dictionary .

Class TunnelPart

Definition:

Subtype Of: [AbstractTunnel](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TunnelPart](#)

Target Role:

Target Class: [AbstractTunnel](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Tunnel](#)

Target Role: tunnelPart

Target Class: [TunnelPart](#)

Attributes

12.9.22. Class TunnelUsageValue

Subclass of <-->

Requirement 117	/req/Tunnel/TunnelUsageValue
A	Any use of the TunnelUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TunnelUsageValue UML class as documented in the TunnelUsageValue section of the Tunnel Data Dictionary .

Class TunnelUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.9.23. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.10. Bridge Model

Requirements Class

<http://www.opengis.net/spec/CityGML/3.0/req/req-class-bridge>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Bridge Model is depicted in [Bridge UML Diagram](#). The Data Dictionary for the Bridge Model Package is provided in section [Bridge Model Data Dictionary](#).

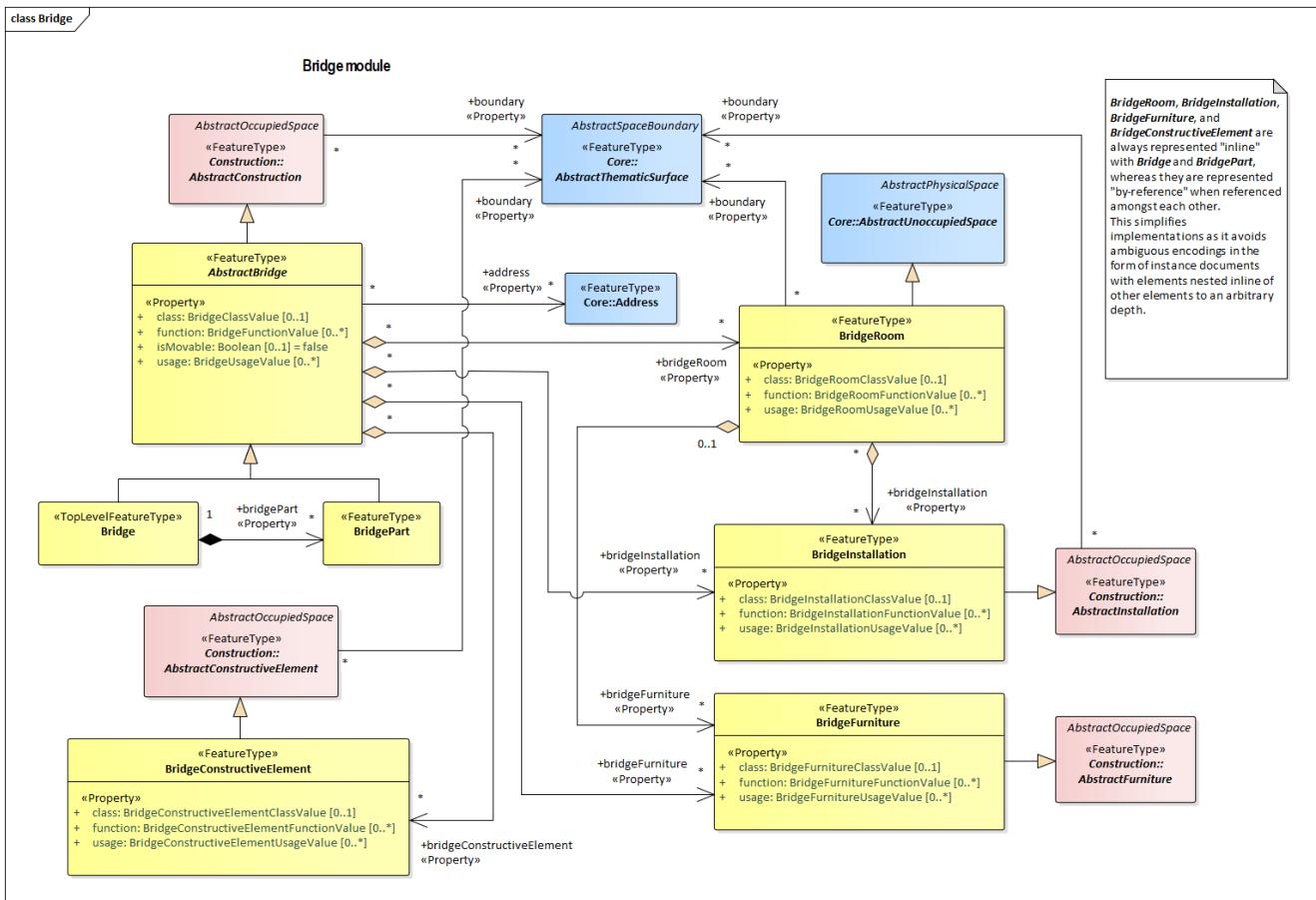


Figure 19. UML diagram of the Bridge Model.

The [UML diagram of the Bridge Model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.11. Bridge Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.11.1. Class AbstractBridge

Subclass of [AbstractConstruction](#)

Requirement 118	/req/Bridge/AbstractBridge
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractBridge UML class as documented in the AbstractBridge section of the Bridge Data Dictionary .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractBridge UML class as documented in the AbstractBridge section of the Bridge Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractBridge UML class; including the name, definition, type, and cardinality of those documented in the AbstractBridge section of the Bridge Data Dictionary .

Class AbstractBridge

Definition:

Subtype Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractBridge](#)

Target Role:

Target Class: [AbstractConstruction](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBridge](#)

Target Role: bridgeRoom

Target Class: [BridgeRoom](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBridge](#)

Target Role: address

Target Class: [Address](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBridge](#)

Target Role: bridgeConstructiveElement

Target Class: [BridgeConstructiveElement](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBridge</p> <p>Target Role: bridgeInstallation</p> <p>Target Class: BridgeInstallation</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBridge</p> <p>Target Role: bridgeFurniture</p> <p>Target Class: BridgeFurniture</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Bridge</p> <p>Target Role:</p> <p>Target Class: AbstractBridge</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BridgePart</p> <p>Target Role:</p> <p>Target Class: AbstractBridge</p>
<p>Attributes</p> <p>Attribute Name: class</p> <p>Value Type: BridgeClassValue</p> <p>Definition: SIG3D: Classification of the actual usage of Bridge or BridgePart as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p> <p>Attribute Name: function</p> <p>Value Type: BridgeFunctionValue</p> <p>Definition: SIG3D: Specified function of Bridge or BridgePart as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

Attribute Name:	isMovable
Value Type:	Boolean
Definition:	SIG3D: Indicates whether a bridge is movable to allow passages for ships (e.g. turnable bridge, liftable bridge)
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeUsageValue
Definition:	SIG3D: Actual usage of Bridge or BridgePart as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.11.2. Class Bridge

Subclass of [AbstractBridge](#)

Requirement 119	/req/Bridge/Bridge
A	The Implementation Specification SHALL contain an element with the same definition as the Bridge UML class as documented in the Bridge section of the Bridge Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Bridge UML class as documented in the Bridge section of the Bridge Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Bridge UML class; including the name, definition, type, and cardinality of those documented in the Bridge section of the Bridge Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Bridge UML class; including the name, definition, type, and cardinality of those documented in the Bridge section of the Bridge Data Dictionary .

Class Bridge

Definition:

Subtype Of: [AbstractBridge](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Bridge
Target Role:	bridgePart
Target Class:	BridgePart
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Bridge
Target Role:	
Target Class:	AbstractBridge
Attributes	

12.11.3. Class BridgeClassValue

Subclass of <-- section,>>

Requirement 120	/req/Bridge/BridgeClassValue
A	Any use of the BridgeClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeClassValue UML class as documented in the BridgeClassValue section of the Bridge Data Dictionary .

Class BridgeClassValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»
Associations
Attributes

12.11.4. Class BridgeConstructiveElement

Subclass of [AbstractConstructiveElement](#)

Requirement 121	/req/Bridge/BridgeConstructiveElement
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeConstructiveElement UML class as documented in the BridgeConstructiveElement section of the Bridge Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeConstructiveElement UML class as documented in the BridgeConstructiveElement section of the Bridge Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BridgeConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BridgeConstructiveElement section of the Bridge Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the BridgeConstructiveElement section of the Bridge Data Dictionary .

Class BridgeConstructiveElement

Definition:

Subtype Of: [AbstractConstructiveElement](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BridgeConstructiveElement](#)

Target Role:

Target Class: [AbstractConstructiveElement](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractBridge](#)

Target Role: bridgeConstructiveElement

Target Class: [BridgeConstructiveElement](#)

Attributes

Attribute Name: class

Value Type: [BridgeConstructiveElementClassValue](#)

Definition: SIG3D: Classification of BridgeConstructionElement as given by the relevant national regulations, information communities, or specific applications.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	BridgeConstructiveElementFunctionValue
Definition:	SIG3D: Specified function of BridgeConstructionElement as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeConstructiveElementUsageValue
Definition:	SIG3D: Actual usage of BridgeConstructionElement as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.11.5. Class BridgeConstructiveElementClassValue

Subclass of <-- section,>>

Requirement 122	/req/Bridge/BridgeConstructiveElementClassValue
A	Any use of the BridgeConstructiveElementClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeConstructiveElementClassValue UML class as documented in the BridgeConstructiveElementClassValue section of the Bridge Data Dictionary .

Class BridgeConstructiveElementClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.6. Class BridgeConstructiveElementFunctionValue

Subclass of <-- section,>>

Requirement 123	/req/Bridge/BridgeConstructiveElementFunctionValue
A	Any use of the BridgeConstructiveElementFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeConstructiveElementFunctionValue UML class as documented in the BridgeConstructiveElementFunctionValue section of the Bridge Data Dictionary .

Class BridgeConstructiveElementFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.7. Class BridgeConstructiveElementUsageValue

Subclass of <-- section,>>

Requirement 124	/req/Bridge/BridgeConstructiveElementUsageValue
A	Any use of the BridgeConstructiveElementUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeConstructiveElementUsageValue UML class as documented in the BridgeConstructiveElementUsageValue section of the Bridge Data Dictionary .

Class BridgeConstructiveElementUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.8. Class BridgeFunctionValue

Subclass of <-- section,>>

Requirement 125	/req/Bridge/BridgeFunctionValue
A	Any use of the BridgeFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFunctionValue UML class as documented in the BridgeFunctionValue section of the Bridge Data Dictionary .

Class BridgeFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.9. Class BridgeFurniture

Subclass of [AbstractFurniture](#)

Requirement 126	/req/Bridge/BridgeFurniture
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeFurniture UML class as documented in the BridgeFurniture section of the Bridge Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeFurniture UML class as documented in the BridgeFurniture section of the Bridge Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BridgeFurniture UML class; including the name, definition, type, and cardinality of those documented in the BridgeFurniture section of the Bridge Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeFurniture UML class; including the name, definition, type, and cardinality of those documented in the BridgeFurniture section of the Bridge Data Dictionary .

Class BridgeFurniture

Definition:

Subtype Of: [AbstractFurniture](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BridgeFurniture](#)

Target Role:

Target Class: [AbstractFurniture](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	BridgeRoom
Target Role:	bridgeFurniture
Target Class:	BridgeFurniture
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBridge
Target Role:	bridgeFurniture
Target Class:	BridgeFurniture
Attributes	
Attribute Name:	class
Value Type:	BridgeFurnitureClassValue
Definition:	SIG3D: Classification of BridgeFurniture as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BridgeFurnitureFunctionValue
Definition:	SIG3D: Specified function of BridgeFurniture as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeFurnitureUsageValue
Definition:	SIG3D: Actual usage of BridgeFurniture as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.11.10. Class BridgeFurnitureClassValue

Subclass of <– section,>>

Requirement 127	/req/Bridge/BridgeFurnitureClassValue
A	Any use of the BridgeFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFurnitureClassValue UML class as documented in the BridgeFurnitureClassValue section of the Bridge Data Dictionary .

Class BridgeFurnitureClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.11. Class BridgeFurnitureFunctionValue

Subclass of <-- section,>>

Requirement 128	/req/Bridge/BridgeFurnitureFunctionValue
A	Any use of the BridgeFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFurnitureFunctionValue UML class as documented in the BridgeFurnitureFunctionValue section of the Bridge Data Dictionary .

Class BridgeFurnitureFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.12. Class BridgeFurnitureUsageValue

Subclass of <-- section,>>

Requirement 129	/req/Bridge/BridgeFurnitureUsageValue
A	Any use of the BridgeFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeFurnitureUsageValue UML class as documented in the BridgeFurnitureUsageValue section of the Bridge Data Dictionary .

Class BridgeFurnitureUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.13. Class BridgeInstallation

Subclass of [AbstractInstallation](#)

Requirement 130	/req/Bridge/BridgeInstallation
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeInstallation UML class as documented in the BridgeInstallation section of the Bridge Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeInstallation UML class as documented in the BridgeInstallation section of the Bridge Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BridgeInstallation UML class; including the name, definition, type, and cardinality of those documented in the BridgeInstallation section of the Bridge Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeInstallation UML class; including the name, definition, type, and cardinality of those documented in the BridgeInstallation section of the Bridge Data Dictionary .

Class BridgeInstallation

Definition:

Subtype Of: [AbstractInstallation](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BridgeInstallation](#)

Target Role:

Target Class: [AbstractInstallation](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	BridgeRoom
Target Role:	bridgeInstallation
Target Class:	BridgeInstallation
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractBridge
Target Role:	bridgeInstallation
Target Class:	BridgeInstallation
Attributes	
Attribute Name:	class
Value Type:	BridgeInstallationClassValue
Definition:	SIG3D: Classification of BridgeInstallation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BridgeInstallationFunctionValue
Definition:	SIG3D: Specified function of BridgeInstallation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeInstallationUsageValue
Definition:	SIG3D: Actual usage of BridgeInstallation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.11.14. Class BridgeInstallationClassValue

Subclass of <– section,>>

Requirement 131	/req/Bridge/BridgeInstallationClassValue
A	Any use of the BridgeInstallationClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationClassValue UML class as documented in the BridgeInstallationClassValue section of the Bridge Data Dictionary .

Class BridgeInstallationClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.15. Class BridgeInstallationFunctionValue

Subclass of <-- section,>>

Requirement 132	/req/Bridge/BridgeInstallationFundtionValue
A	Any use of the BridgeInstallationFundtionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationFundtionValue UML class as documented in the BridgeInstallationFundtionValue section of the Bridge Data Dictionary .

Class BridgeInstallationFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.16. Class BridgeInstallationUsageValue

Subclass of <-- section,>>

Requirement 133	/req/Bridge/BridgeInstallationUsageValue
A	Any use of the BridgeInstallationUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationUsageValue UML class as documented in the BridgeInstallationUsageValue section of the Bridge Data Dictionary .

Class BridgeInstallationUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.17. Class BridgePart

Subclass of [AbstractBridge](#)

Requirement 134	/req/Bridge/BridgePart
A	The Implementation Specification SHALL contain an element with the same definition as the BridgePart UML class as documented in the BridgePart section of the Bridge Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgePart UML class as documented in the BridgePart section of the Bridge Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BridgePart UML class; including the name, definition, type, and cardinality of those documented in the BridgePart section of the Bridge Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgePart UML class; including the name, definition, type, and cardinality of those documented in the BridgePart section of the Bridge Data Dictionary .

Class BridgePart

Definition:

Subtype Of: [AbstractBridge](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BridgePart](#)

Target Role:

Target Class: [AbstractBridge](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Bridge](#)

Target Role: bridgePart

Target Class: [BridgePart](#)

Attributes

12.11.18. Class BridgeRoom

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 135	/req/Bridge/BridgeRoom
A	The Implementation Specification SHALL contain an element with the same definition as the BridgeRoom UML class as documented in the BridgeRoom section of the Bridge Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the BridgeRoom UML class as documented in the BridgeRoom section of the Bridge Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the BridgeRoom UML class; including the name, definition, type, and cardinality of those documented in the BridgeRoom section of the Bridge Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the BridgeRoom UML class; including the name, definition, type, and cardinality of those documented in the BridgeRoom section of the Bridge Data Dictionary .

Class BridgeRoom

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [BridgeRoom](#)

Target Role: bridgeInstallation

Target Class: [BridgeInstallation](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [BridgeRoom](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BridgeRoom</p> <p>Target Role:</p> <p>Target Class: AbstractUnoccupiedSpace</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: BridgeRoom</p> <p>Target Role: bridgeFurniture</p> <p>Target Class: BridgeFurniture</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBridge</p> <p>Target Role: bridgeRoom</p> <p>Target Class: BridgeRoom</p>
<p>Attributes</p> <p>Attribute Name: class</p> <p>Value Type: BridgeRoomClassValue</p> <p>Definition: SIG3D: Classification of BridgeRoom as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: function</p> <p>Value Type: BridgeRoomFunctionValue</p> <p>Definition: SIG3D: Specified function of BridgeRoom as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: usage</p> <p>Value Type: BridgeRoomUsageValue</p> <p>Definition: SIG3D: Actual usage of BridgeRoom as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

12.11.19. Class BridgeRoomClassValue

Subclass of <← section,>>

Requirement 136	/req/Bridge/BridgeRoomClassValue
A	Any use of the BridgeRoomClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeRoomClassValue UML class as documented in the BridgeRoomClassValue section of the Bridge Data Dictionary .

Class BridgeRoomClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.20. Class BridgeRoomFunctionValue

Subclass of <-- section,>>

Requirement 137	/req/Bridge/BridgeRoomFunctionValue
A	Any use of the BridgeRoomFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeRoomFunctionValue UML class as documented in the BridgeRoomFunctionValue section of the Bridge Data Dictionary .

Class BridgeRoomFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.21. Class BridgeRoomUsageValue

Subclass of <-- section,>>

Requirement 138	/req/Bridge/BridgeRoomUsageValue
A	Any use of the BridgeRoomUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeRoomUsageValue UML class as documented in the BridgeRoomUsageValue section of the Bridge Data Dictionary .

Class BridgeRoomUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.22. Class BridgeUsageValue

Subclass of <-- section,>>

Requirement 139	/req/Bridge/BridgeUsageValue
A	Any use of the BridgeUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeUsageValue UML class as documented in the BridgeUsageValue section of the Bridge Data Dictionary .

Class BridgeUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.11.23. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.12. Water Body

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-waterbody>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Water Body Model is depicted in [Water Body UML Diagram](#). The Data Dictionary for the Water Body Package is provided in section [Water Body Data Dictionary](#).

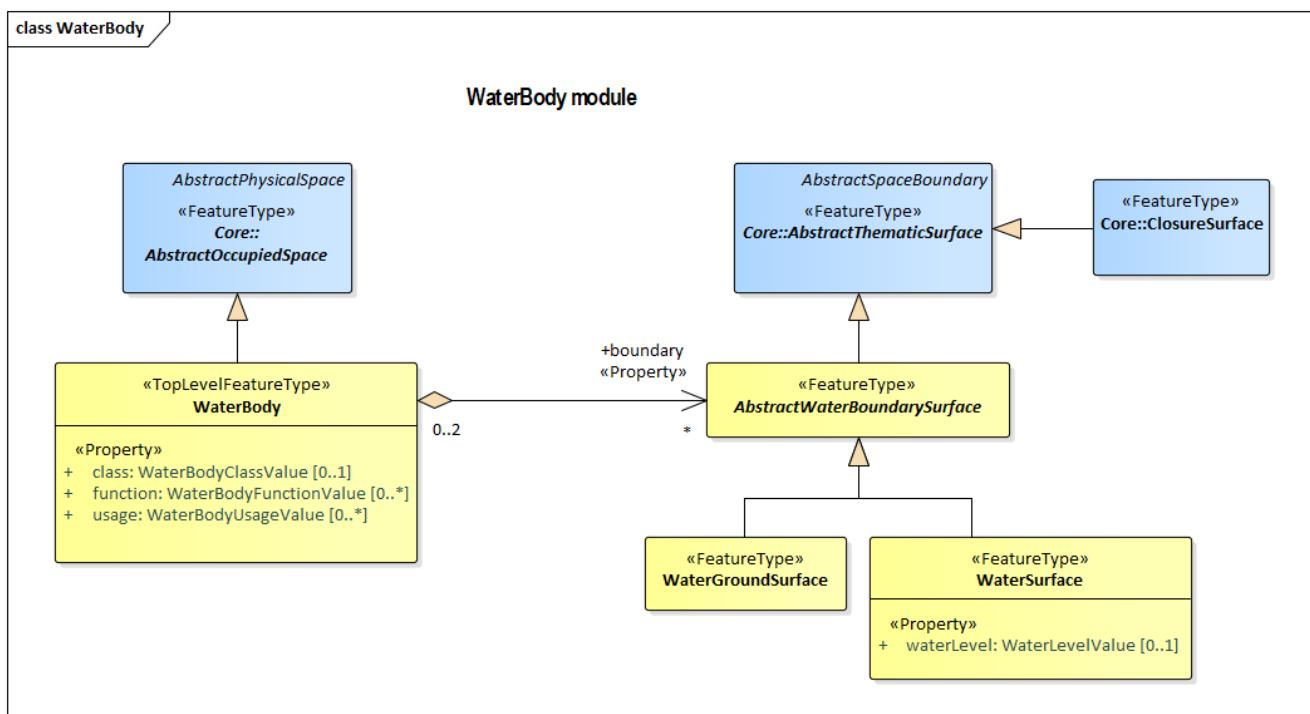


Figure 20. UML diagram of the Water Body Model.

The [UML diagram of the Water Body Model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.13. WaterBody Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.13.1. Class AbstractWaterBoundarySurface

Subclass of [AbstractThematicSurface](#)

Requirement 140	/req/Waterbody/AbstractWaterBoundarySurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractWaterBoundarySurface UML class as documented in the AbstractWaterBoundarySurface section of the Waterbody Data Dictionary .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractWaterBoundarySurface UML class as documented in the AbstractWaterBoundarySurface section of the Waterbody Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractWaterBoundarySurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractWaterBoundarySurface section of the Waterbody Data Dictionary .

Class AbstractWaterBoundarySurface

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractWaterBoundarySurface](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [WaterSurface](#)

Target Role:

Target Class: [AbstractWaterBoundarySurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [WaterBody](#)

Target Role: boundary

Target Class: [AbstractWaterBoundarySurface](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	WaterGroundSurface
Target Role:	
Target Class:	AbstractWaterBoundarySurface
Attributes	

12.13.2. Class WaterBody

Subclass of [AbstractOccupiedSpace](#)

Requirement 141	/req/Waterbody/WaterBody
A	The Implementation Specification SHALL contain an element with the same definition as the WaterBody UML class as documented in the WaterBody section of the Waterbody Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WaterBody UML class as documented in the WaterBody section of the Waterbody Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the WaterBody UML class; including the name, definition, type, and cardinality of those documented in the WaterBody section of the Waterbody Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WaterBody UML class; including the name, definition, type, and cardinality of those documented in the WaterBody section of the Waterbody Data Dictionary .

Class WaterBody	
Definition:	
Subtype Of: AbstractOccupiedSpace	
Stereotype: «TopLevelFeatureType»	
Associations	
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	WaterBody
Target Role:	
Target Class:	AbstractOccupiedSpace

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	WaterBody
Target Role:	boundary
Target Class:	AbstractWaterBoundarySurface
Attributes	
Attribute Name:	class
Value Type:	WaterBodyClassValue
Definition:	SIG3D: Classification of WaterBody as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	WaterBodyFunctionValue
Definition:	SIG3D: Specified function of WaterBody as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	WaterBodyUsageValue
Definition:	SIG3D: Actual usage of WaterBody as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.13.3. Class WaterBodyClassValue

Subclass of <– section,>>

Requirement 142	/req/Waterbody/WaterBodyClassValue
A	Any use of the WaterBodyClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterBodyClassValue UML class as documented in the WaterBodyClassValue section of the Waterbody Data Dictionary .

Class WaterBodyClassValue
Definition:
Subtype Of: <– section,>>
Stereotype: «CodeList»
Associations
Attributes

12.13.4. Class WaterBodyFunctionValue

Subclass of <-- section,>>

Requirement 143	/req/Waterbody/WaterBodyFunctionValue
A	Any use of the WaterBodyFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterBodyFunctionValue UML class as documented in the WaterBodyFunctionValue section of the Waterbody Data Dictionary .

Class WaterBodyFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.13.5. Class WaterBodyUsageValue

Subclass of <-- section,>>

Requirement 144	/req/Waterbody/WaterBodyUsageValue
A	Any use of the WaterBodyUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterBodyUsageValue UML class as documented in the WaterBodyUsageValue section of the Waterbody Data Dictionary .

Class WaterBodyUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.13.6. Class WaterGroundSurface

Subclass of [AbstractWaterBoundarySurface](#)

Requirement 145	/req/Waterbody/WaterGroundSurface

A	The Implementation Specification SHALL contain an element with the same definition as the WaterGroundSurface UML class as documented in the WaterGroundSurface section of the Waterbody Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WaterGroundSurface UML class as documented in the WaterGroundSurface section of the Waterbody Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the WaterGroundSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterGroundSurface section of the Waterbody Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WaterGroundSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterGroundSurface section of the Waterbody Data Dictionary .

Class WaterGroundSurface

Definition:

Subtype Of: [AbstractWaterBoundarySurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [WaterGroundSurface](#)

Target Role:

Target Class: [AbstractWaterBoundarySurface](#)

Attributes

12.13.7. Class WaterLevelValue

Subclass of <– section,>>

Requirement 146	/req/Waterbody/WaterLevelValue
A	Any use of the WaterLevelValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterLevelValue UML class as documented in the WaterLevelValue section of the Waterbody Data Dictionary .

Class WaterLevelValue

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «CodeList»

Associations

Attributes

12.13.8. Class WaterSurface

Subclass of [AbstractWaterBoundarySurface](#)

Requirement 147	/req/Waterbody/WaterSurface
A	The Implementation Specification SHALL contain an element with the same definition as the WaterSurface UML class as documented in the WaterSurface section of the Waterbody Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WaterSurface UML class as documented in the WaterSurface section of the Waterbody Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the WaterSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterSurface section of the Waterbody Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WaterSurface UML class; including the name, definition, type, and cardinality of those documented in the WaterSurface section of the Waterbody Data Dictionary .

Class WaterSurface

Definition:
 Subtype Of: [AbstractWaterBoundarySurface](#)
 Stereotype: «FeatureType»

Associations

Name:
 Type: Generalization
 Direction: Source → Destination
 Source Role:
 Source Class: [WaterSurface](#)
 Target Role:
 Target Class: [AbstractWaterBoundarySurface](#)

Attributes

Attribute Name:	waterLevel
Value Type:	WaterLevelValue
Definition:	SIG3D: Codelist of the WaterSurface property waterLevel.
Multiplicity:	[0..1]
Stereotype:	«Property»

12.13.9. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.14. Transportation

Requirements Class	
http://www.opengis.net/spec/CityGML/3.0/req/req-class-transportation	
Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Transportation Model is depicted in [Transportation UML Diagram](#). The Data Dictionary for the Transportation Package is provided in section [Transportation Model Data Dictionary](#).

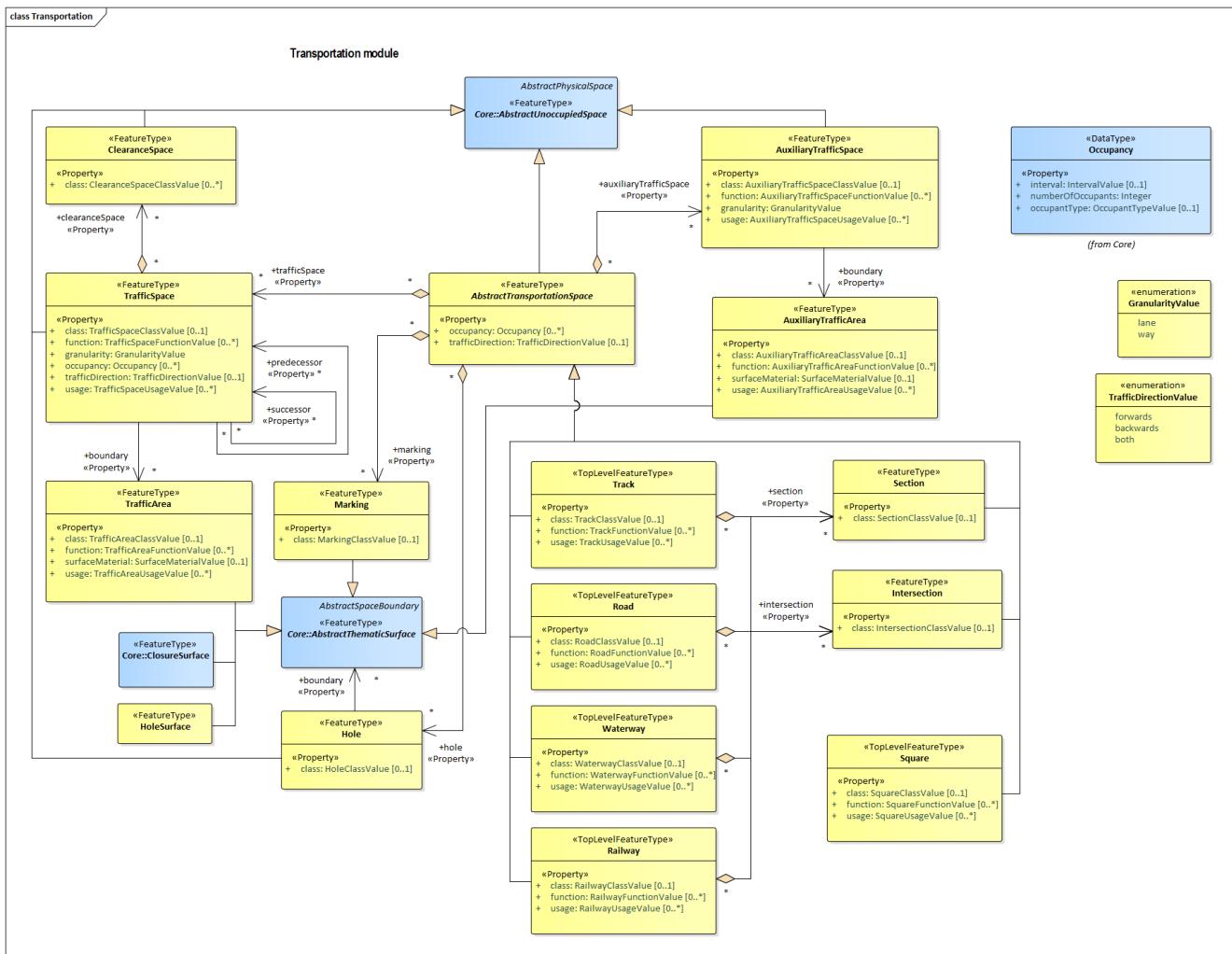


Figure 21. UML diagram of the Transportation Model.

The [UML diagram of the Transportation Model](#). is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.15. Transportation Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.15.1. Class AbstractTransportationSpace

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 148	/req/Transportation/AbstractTransportationSpace
-----------------	---

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTransportationSpace UML class as documented in the AbstractTransportationSpace section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTransportationSpace UML class as documented in the AbstractTransportationSpace section of the Transportation Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTransportationSpace UML class; including the name, definition, type, and cardinality of those documented in the AbstractTransportationSpace section of the Transportation Data Dictionary .

Class AbstractTransportationSpace

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTransportationSpace](#)

Target Role: trafficSpace

Target Class: [TrafficSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTransportationSpace](#)

Target Role: hole

Target Class: [Hole](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTransportationSpace](#)

Target Role: auxiliaryTrafficSpace

Target Class: [AuxiliaryTrafficSpace](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractTransportationSpace
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractTransportationSpace
Target Role:	marking
Target Class:	Marking
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Intersection
Target Role:	
Target Class:	AbstractTransportationSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Waterway
Target Role:	
Target Class:	AbstractTransportationSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Track
Target Role:	
Target Class:	AbstractTransportationSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Section
Target Role:	
Target Class:	AbstractTransportationSpace

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Road
Target Role:	
Target Class:	AbstractTransportationSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Railway
Target Role:	
Target Class:	AbstractTransportationSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Square
Target Role:	
Target Class:	AbstractTransportationSpace
Attributes	
Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	trafficDirection
Value Type:	TrafficDirectionValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.15.2. Class AuxiliaryTrafficArea

Subclass of [AbstractThematicSurface](#)

Requirement 149	/req/Transportation/AuxiliaryTrafficArea
A	The Implementation Specification SHALL contain an element with the same definition as the AuxiliaryTrafficArea UML class as documented in the AuxiliaryTrafficArea section of the Transportation Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the AuxiliaryTrafficArea UML class as documented in the AuxiliaryTrafficArea section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the AuxiliaryTrafficArea UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficArea section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the AuxiliaryTrafficArea UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficArea section of the Transportation Data Dictionary .

Class AuxiliaryTrafficArea

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AuxiliaryTrafficArea](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AuxiliaryTrafficSpace](#)

Target Role: boundary

Target Class: [AuxiliaryTrafficArea](#)

Attributes

Attribute Name: class

Value Type: [AuxiliaryTrafficAreaClassValue](#)

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	AuxiliaryTrafficAreaFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	surfaceMaterial
Value Type:	SurfaceMaterialValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	AuxiliaryTrafficAreaUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.3. Class AuxiliaryTrafficAreaClassValue

Subclass of <-- section,>>

Requirement 150	/req/Transportation/AuxiliaryTrafficAreaClassValue
A	Any use of the AuxiliaryTrafficAreaClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficAreaClassValue UML class as documented in the AuxiliaryTrafficAreaClassValue section of the Transportation Data Dictionary .

Class AuxiliaryTrafficAreaClassValue

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «CodeList»

Associations

Attributes

12.15.4. Class AuxiliaryTrafficAreaFunctionValue

Subclass of <-- section,>>

Requirement 151	/req/Transportation/AuxiliaryTrafficAreaFunctionValue

A	Any use of the AuxiliaryTrafficAreaFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficAreaFunctionValue UML class as documented in the AuxiliaryTrafficAreaFunctionValue section of the Transportation Data Dictionary .
---	--

Class AuxiliaryTrafficAreaFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.5. Class AuxiliaryTrafficAreaUsageValue

Subclass of <-- section,>>

Requirement 152	/req/Transportation/AuxiliaryTrafficAreaUsageValue
A	Any use of the AuxiliaryTrafficAreaUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficAreaUsageValue UML class as documented in the AuxiliaryTrafficAreaUsageValue section of the Transportation Data Dictionary .

Class AuxiliaryTrafficAreaUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.6. Class AuxiliaryTrafficSpace

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 153	/req/Transportation/AuxiliaryTrafficSpace
A	The Implementation Specification SHALL contain an element with the same definition as the AuxiliaryTrafficSpace UML class as documented in the AuxiliaryTrafficSpace section of the Transportation Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the AuxiliaryTrafficSpace UML class as documented in the AuxiliaryTrafficSpace section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the AuxiliaryTrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficSpace section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the AuxiliaryTrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the AuxiliaryTrafficSpace section of the Transportation Data Dictionary .

Class AuxiliaryTrafficSpace

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AuxiliaryTrafficSpace](#)

Target Role:

Target Class: [AbstractUnoccupiedSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AuxiliaryTrafficSpace](#)

Target Role: boundary

Target Class: [AuxiliaryTrafficArea](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractTransportationSpace](#)

Target Role: auxiliaryTrafficSpace

Target Class: [AuxiliaryTrafficSpace](#)

Attributes

Attribute Name:	class
Value Type:	AuxiliaryTrafficSpaceClassValue
Definition:	SIG3D: Classification of AuxiliaryTrafficSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	AuxiliaryTrafficSpaceFunctionValue
Definition:	SIG3D: Specified function of AuxiliaryTrafficSpace given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	granularity
Value Type:	GranularityValue
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	AuxiliaryTrafficSpaceUsageValue
Definition:	SIG3D: Actual usage of AuxiliaryTrafficSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.7. Class AuxiliaryTrafficSpaceClassValue

Subclass of <-- section,>>

Requirement 154	/req/Transportation/AuxiliaryTrafficSpaceClassValue
A	Any use of the AuxiliaryTrafficSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficSpaceClassValue UML class as documented in the AuxiliaryTrafficSpaceClassValue section of the Transportation Data Dictionary .

Class AuxiliaryTrafficSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.8. Class AuxiliaryTrafficSpaceFunctionValue

Subclass of <-- section,>>

Requirement 155	/req/Transportation/AuxiliaryTrafficSpaceFunctionValue
A	Any use of the AuxiliaryTrafficSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficSpaceFunctionValue UML class as documented in the AuxiliaryTrafficSpaceFunctionValue section of the Transportation Data Dictionary .

Class AuxiliaryTrafficSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.9. Class AuxiliaryTrafficSpaceUsageValue

Subclass of <-- section,>>

Requirement 156	/req/Transportation/AuxiliaryTrafficSpaceUsageValue
A	Any use of the AuxiliaryTrafficSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuxiliaryTrafficSpaceUsageValue UML class as documented in the AuxiliaryTrafficSpaceUsageValue section of the Transportation Data Dictionary .

Class AuxiliaryTrafficSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.10. Class ClearanceSpace

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 157	/req/Transportation/ClearanceSpace
-----------------	------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the ClearanceSpace UML class as documented in the ClearanceSpace section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ClearanceSpace UML class as documented in the ClearanceSpace section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ClearanceSpace UML class; including the name, definition, type, and cardinality of those documented in the ClearanceSpace section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ClearanceSpace UML class; including the name, definition, type, and cardinality of those documented in the ClearanceSpace section of the Transportation Data Dictionary .

Class ClearanceSpace

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [ClearanceSpace](#)

Target Role:

Target Class: [AbstractUnoccupiedSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TrafficSpace](#)

Target Role: clearanceSpace

Target Class: [ClearanceSpace](#)

Attributes

Attribute Name: class

Value Type: ClearanceSpaceClassValue

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

12.15.11. Class ClearanceSpaceClassValue

Subclass of <-- section,>>

Requirement 158	/req/Transportation/ClearanceSpaceClassValue
A	Any use of the ClearanceSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the ClearanceSpaceClassValue UML class as documented in the ClearanceSpaceClassValue section of the Transportation Data Dictionary .

Class ClearanceSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.12. Class Hole

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 159	/req/Transportation/Hole
A	The Implementation Specification SHALL contain an element with the same definition as the Hole UML class as documented in the Hole section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Hole UML class as documented in the Hole section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Hole UML class; including the name, definition, type, and cardinality of those documented in the Hole section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Hole UML class; including the name, definition, type, and cardinality of those documented in the Hole section of the Transportation Data Dictionary .

Class Hole

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations	
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Hole
Target Role:	
Target Class:	AbstractUnoccupiedSpace
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Hole
Target Role:	boundary
Target Class:	AbstractThematicSurface
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractTransportationSpace
Target Role:	hole
Target Class:	Hole
Attributes	
Attribute Name:	class
Value Type:	HoleClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.15.13. Class HoleClassValue

Subclass of <– section,>>

Requirement 160	/req/Transportation/HoleClassValue
A	Any use of the HoleClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HoleClassValue UML class as documented in the HoleClassValue section of the Transportation Data Dictionary .

Class HoleClassValue	
Definition:	
Subtype Of:	<– section,>>
Stereotype:	«CodeList»

Associations	
Name:	
Type:	NoteLink
Direction:	Source → Destination
Source Role:	
Source Class:	Note
Target Role:	
Target Class:	HoleClassValue
Attributes	

12.15.14. Class HoleSurface

Subclass of [AbstractThematicSurface](#)

Requirement 161	/req/Transportation/HoleSurface
A	The Implementation Specification SHALL contain an element with the same definition as the HoleSurface UML class as documented in the HoleSurface section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the HoleSurface UML class as documented in the HoleSurface section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the HoleSurface UML class; including the name, definition, type, and cardinality of those documented in the HoleSurface section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the HoleSurface UML class; including the name, definition, type, and cardinality of those documented in the HoleSurface section of the Transportation Data Dictionary .

Class HoleSurface	
Definition:	
Subtype Of:	AbstractThematicSurface
Stereotype:	«FeatureType»
Associations	
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	HoleSurface
Target Role:	
Target Class:	AbstractThematicSurface

Attributes

12.15.15. Class Intersection

Subclass of [AbstractTransportationSpace](#)

Requirement 162	/req/Transportation/Intersection
A	The Implementation Specification SHALL contain an element with the same definition as the Intersection UML class as documented in the Intersection section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Intersection UML class as documented in the Intersection section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Intersection UML class; including the name, definition, type, and cardinality of those documented in the Intersection section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Intersection UML class; including the name, definition, type, and cardinality of those documented in the Intersection section of the Transportation Data Dictionary .

Class Intersection

Definition:

Subtype Of: [AbstractTransportationSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Intersection](#)

Target Role:

Target Class: [AbstractTransportationSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Waterway](#)

Target Role: intersection

Target Class: [Intersection](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Road</p> <p>Target Role: intersection</p> <p>Target Class: Intersection</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Railway</p> <p>Target Role: intersection</p> <p>Target Class: Intersection</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Track</p> <p>Target Role: intersection</p> <p>Target Class: Intersection</p>
<p>Attributes</p> <p>Attribute Name: class</p> <p>Value Type: IntersectionClassValue</p> <p>Definition:</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

12.15.16. Class IntersectionClassValue

Subclass of <-- section,>>

Requirement 163	/req/Transportation/IntersectionClassValue
A	Any use of the IntersectionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the IntersectionClassValue UML class as documented in the IntersectionClassValue section of the Transportation Data Dictionary .

Class IntersectionClassValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»

Associations
Attributes

12.15.17. Class Marking

Subclass of [AbstractThematicSurface](#)

Requirement 164	/req/Transportation/Marking
A	The Implementation Specification SHALL contain an element with the same definition as the Marking UML class as documented in the Marking section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Marking UML class as documented in the Marking section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Marking UML class; including the name, definition, type, and cardinality of those documented in the Marking section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Marking UML class; including the name, definition, type, and cardinality of those documented in the Marking section of the Transportation Data Dictionary .

Class Marking
Definition:
Subtype Of: AbstractThematicSurface
Stereotype: «FeatureType»
Associations
Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: Marking
Target Role:
Target Class: AbstractThematicSurface
Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: AbstractTransportationSpace
Target Role: marking
Target Class: Marking

Attributes

Attribute Name: class
Value Type: MarkingClassValue
Definition:
Multiplicity: [0..1]
Stereotype: «Property»

12.15.18. Class MarkingClassValue

Subclass of <-- section,>>

Requirement 165	/req/Transportation/MarkingClassValue
A	Any use of the MarkingClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the MarkingClassValue UML class as documented in the MarkingClassValue section of the Transportation Data Dictionary .

Class MarkingClassValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.15.19. Class Railway

Subclass of [AbstractTransportationSpace](#)

Requirement 166	/req/Transportation/Railway
A	The Implementation Specification SHALL contain an element with the same definition as the Railway UML class as documented in the Railway section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Railway UML class as documented in the Railway section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Railway UML class; including the name, definition, type, and cardinality of those documented in the Railway section of the Transportation Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the Railway UML class; including the name, definition, type, and cardinality of those documented in the Railway section of the Transportation Data Dictionary .
---	--

Class Railway

Definition:

Subtype Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Railway](#)

Target Role: section

Target Class: [Section](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Railway](#)

Target Role: intersection

Target Class: [Intersection](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Railway](#)

Target Role:

Target Class: [AbstractTransportationSpace](#)

Attributes

Attribute Name: class

Value Type: RailwayClassValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: RailwayFunctionValue

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

Attribute Name:	usage
Value Type:	RailwayUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.20. Class RailwayClassValue

Subclass of <-- section,>>

Requirement 167	/req/Transportation/RailwayClassValue
A	Any use of the RailwayClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RailwayClassValue UML class as documented in the RailwayClassValue section of the Transportation Data Dictionary .

Class RailwayClassValue

Definition:

Subtype Of: <-- section,>>

StereoType: «CodeList»

Associations

Attributes

12.15.21. Class RailwayFunctionValue

Subclass of <-- section,>>

Requirement 168	/req/Transportation/RailwayFunctionValue
A	Any use of the RailwayFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RailwayFunctionValue UML class as documented in the RailwayFunctionValue section of the Transportation Data Dictionary .

Class RailwayFunctionValue

Definition:

Subtype Of: <-- section,>>

StereoType: «CodeList»

Associations

Attributes

12.15.22. Class RailwayUsageValue

Subclass of <-- section,>>

Requirement 169	/req/Transportation/RailwayUsageValue
A	Any use of the RailwayUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RailwayUsageValue UML class as documented in the RailwayUsageValue section of the Transportation Data Dictionary .

Class RailwayUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.23. Class Road

Subclass of [AbstractTransportationSpace](#)

Requirement 170	/req/Transportation/Road
A	The Implementation Specification SHALL contain an element with the same definition as the Road UML class as documented in the Road section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Road UML class as documented in the Road section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Road UML class; including the name, definition, type, and cardinality of those documented in the Road section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Road UML class; including the name, definition, type, and cardinality of those documented in the Road section of the Transportation Data Dictionary .

Class Road

Definition:

Subtype Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

Associations	
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Road
Target Role:	intersection
Target Class:	Intersection
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Road
Target Role:	section
Target Class:	Section
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Road
Target Role:	
Target Class:	AbstractTransportationSpace
Attributes	
Attribute Name:	class
Value Type:	RoadClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	RoadFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	RoadUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.24. Class RoadClassValue

Subclass of <← section,>>

Requirement 171	/req/Transportation/RoadClassValue
-----------------	------------------------------------

A	Any use of the RoadClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoadClassValue UML class as documented in the RoadClassValue section of the Transportation Data Dictionary .
---	---

Class RoadClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.25. Class RoadFunctionValue

Subclass of <-- section,>>

Requirement 172	/req/Transportation/RoadFunctionValue
A	Any use of the RoadFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoadFunctionValue UML class as documented in the RoadFunctionValue section of the Transportation Data Dictionary .

Class RoadFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.26. Class RoadUsageValue

Subclass of <-- section,>>

Requirement 173	/req/Transportation/RoadUsageValue
A	Any use of the RoadUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the RoadUsageValue UML class as documented in the RoadUsageValue section of the Transportation Data Dictionary .

Class RoadUsageValue

Definition:
Subtype Of: <--section,>>
StereoType: «CodeList»

Associations

Attributes

12.15.27. Class Section

Subclass of [AbstractTransportationSpace](#)

Requirement 174	/req/Transportation/Section
A	The Implementation Specification SHALL contain an element with the same definition as the Section UML class as documented in the Section section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Section UML class as documented in the Section section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Section UML class; including the name, definition, type, and cardinality of those documented in the Section section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Section UML class; including the name, definition, type, and cardinality of those documented in the Section section of the Transportation Data Dictionary .

Class Section

Definition:
Subtype Of: [AbstractTransportationSpace](#)
StereoType: «FeatureType»

Associations

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [Section](#)
Target Role:
Target Class: [AbstractTransportationSpace](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Waterway
Target Role:	section
Target Class:	Section
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Railway
Target Role:	section
Target Class:	Section
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Road
Target Role:	section
Target Class:	Section
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Track
Target Role:	section
Target Class:	Section
Attributes	
Attribute Name:	class
Value Type:	SectionClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.15.28. Class SectionClassValue

Subclass of <-- section,>>

Requirement 175	/req/Transportation/SectionClassValue
A	Any use of the SectionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SectionClassValue UML class as documented in the SectionClassValue section of the Transportation Data Dictionary .

Class SectionClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.29. Class Square

Subclass of [AbstractTransportationSpace](#)

Requirement 176	/req/Transportation/Square
A	The Implementation Specification SHALL contain an element with the same definition as the Square UML class as documented in the Square section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Square UML class as documented in the Square section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Square UML class; including the name, definition, type, and cardinality of those documented in the Square section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Square UML class; including the name, definition, type, and cardinality of those documented in the Square section of the Transportation Data Dictionary .

Class Square

Definition:

Subtype Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Square](#)

Target Role:

Target Class: [AbstractTransportationSpace](#)

Attributes

Attribute Name:	class
Value Type:	SquareClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	function
Value Type:	SquareFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

Attribute Name:	usage
Value Type:	SquareUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.30. Class SquareClassValue

Subclass of <-- section,>>

Requirement 177	/req/Transportation/SquareClassValue
A	Any use of the SquareClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SquareClassValue UML class as documented in the SquareClassValue section of the Transportation Data Dictionary .

Class SquareClassValue

Definition:
 Subtype Of: <-- section,>>
 StereoType: «CodeList»

Associations

Attributes

12.15.31. Class SquareFunctionValue

Subclass of <-- section,>>

Requirement 178	/req/Transportation/SquareFunctionValue
A	Any use of the SquareFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SquareFunctionValue UML class as documented in the SquareFunctionValue section of the Transportation Data Dictionary .

Class SquareFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.32. Class SquareUsageValue

Subclass of <-- section,>>

Requirement 179	/req/Transportation/SquareUsageValue
A	Any use of the SquareUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SquareUsageValue UML class as documented in the SquareUsageValue section of the Transportation Data Dictionary .

Class SquareUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.33. Class SurfaceMaterialValue

Subclass of <-- section,>>

Requirement 180	/req/Transportation/SurfaceMaterialValue
A	Any use of the SurfaceMaterialValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SurfaceMaterialValue UML class as documented in the SurfaceMaterialValue section of the Transportation Data Dictionary .

Class SurfaceMaterialValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.34. Class Track

Subclass of [AbstractTransportationSpace](#)

Requirement 181	/req/Transportation/Track
A	The Implementation Specification SHALL contain an element with the same definition as the Track UML class as documented in the Track section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Track UML class as documented in the Track section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Track UML class; including the name, definition, type, and cardinality of those documented in the Track section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Track UML class; including the name, definition, type, and cardinality of those documented in the Track section of the Transportation Data Dictionary .

Class Track

Definition:

Subtype Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Track](#)

Target Role:

Target Class: [AbstractTransportationSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Track](#)

Target Role: section

Target Class: [Section](#)

Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Track
Target Role:	intersection
Target Class:	Intersection
Attributes	
Attribute Name:	class
Value Type:	TrackClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	TrackFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	TrackUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.35. Class TrackClassValue

Subclass of <-- section,>>

Requirement 182	/req/Transportation/TrackClassValue
A	Any use of the TrackClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrackClassValue UML class as documented in the TrackClassValue section of the Transportation Data Dictionary .

Class TrackClassValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»
Associations
Attributes

12.15.36. Class TrackFunctionValue

Subclass of <-- section,>>

Requirement 183	/req/Transportation/TrackFunctionValue
A	Any use of the TrackFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrackFunctionValue UML class as documented in the TrackFunctionValue section of the Transportation Data Dictionary .

Class TrackFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.37. Class TrackUsageValue

Subclass of <-- section,>>

Requirement 184	/req/Transportation/TrackUsageValue
A	Any use of the TrackUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrackUsageValue UML class as documented in the TrackUsageValue section of the Transportation Data Dictionary .

Class TrackUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.38. Class TrafficArea

Subclass of [AbstractThematicSurface](#)

Requirement 185	/req/Transportation/TrafficArea
-----------------	---------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the TrafficArea UML class as documented in the TrafficArea section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TrafficArea UML class as documented in the TrafficArea section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TrafficArea UML class; including the name, definition, type, and cardinality of those documented in the TrafficArea section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TrafficArea UML class; including the name, definition, type, and cardinality of those documented in the TrafficArea section of the Transportation Data Dictionary .

Class TrafficArea

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TrafficArea](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TrafficSpace](#)

Target Role: boundary

Target Class: [TrafficArea](#)

Attributes

Attribute Name: class

Value Type: [TrafficAreaClassValue](#)

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	TrafficAreaFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	surfaceMaterial
Value Type:	SurfaceMaterialValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	TrafficAreaUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.39. Class TrafficAreaClassValue

Subclass of <-- section,>>

Requirement 186	/req/Transportation/TrafficAreaClassValue
A	Any use of the TrafficAreaClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficAreaClassValue UML class as documented in the Transportation Data Dictionary .

Class TrafficAreaClassValue

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «CodeList»

Associations

Attributes

12.15.40. Class TrafficAreaFunctionValue

Subclass of <-- section,>>

Requirement 187	/req/Transportation/TrafficAreaFunctionValue
-----------------	--

A	Any use of the TrafficAreaFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficAreaFunctionValue UML class as documented in the TrafficAreaFunctionValue section of the Transportation Data Dictionary .
---	---

Class TrafficAreaFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.41. Class TrafficAreaUsageValue

Subclass of <-- section,>>

Requirement 188	/req/Transportation/TrafficAreaUsageValue
A	Any use of the TrafficAreaUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficAreaUsageValue UML class as documented in the TrafficAreaUsageValue section of the Transportation Data Dictionary .

Class TrafficAreaUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.42. Class TrafficSpace

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 189	/req/Transportation/TrafficSpace
A	The Implementation Specification SHALL contain an element with the same definition as the TrafficSpace UML class as documented in the TrafficSpace section of the Transportation Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TrafficSpace UML class as documented in the TrafficSpace section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the TrafficSpace section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TrafficSpace UML class; including the name, definition, type, and cardinality of those documented in the TrafficSpace section of the Transportation Data Dictionary .

Class TrafficSpace

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TrafficSpace](#)

Target Role: clearanceSpace

Target Class: [ClearanceSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TrafficSpace](#)

Target Role: successor

Target Class: [TrafficSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TrafficSpace](#)

Target Role: boundary

Target Class: [TrafficArea](#)

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: TrafficSpace</p> <p>Target Role: predecessor</p> <p>Target Class: TrafficSpace</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: TrafficSpace</p> <p>Target Role:</p> <p>Target Class: AbstractUnoccupiedSpace</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractTransportationSpace</p> <p>Target Role: trafficSpace</p> <p>Target Class: TrafficSpace</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: TrafficSpace</p> <p>Target Role: successor</p> <p>Target Class: TrafficSpace</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: TrafficSpace</p> <p>Target Role: predecessor</p> <p>Target Class: TrafficSpace</p>
<p>Attributes</p> <p>Attribute Name: class</p> <p>Value Type: TrafficSpaceClassValue</p> <p>Definition: SIG3D: Classification of TrafficSpace as given by the relevant national regulations, information communities, or specific applications.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

Attribute Name:	function
Value Type:	TrafficSpaceFunctionValue
Definition:	SIG3D: Specified function of TrafficSpace given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	granularity
Value Type:	GranularityValue
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	trafficDirection
Value Type:	TrafficDirectionValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	TrafficSpaceUsageValue
Definition:	SIG3D: Actual usage of TrafficSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.15.43. Class TrafficSpaceClassValue

Subclass of <-- section,>>

Requirement 190	/req/Transportation/TrafficSpaceClassValue
A	Any use of the TrafficSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficSpaceClassValue UML class as documented in the Transportation Data Dictionary .

Class TrafficSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations**Attributes**

12.15.44. Class TrafficSpaceFunctionValue

Subclass of <-- section,>>

Requirement 191	/req/Transportation/TrafficSpaceFunctionValue
A	Any use of the TrafficSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficSpaceFunctionValue UML class as documented in the TrafficSpaceFunctionValue section of the Transportation Data Dictionary .

Class TrafficSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations**Attributes**

12.15.45. Class TrafficSpaceUsageValue

Subclass of <-- section,>>

Requirement 192	/req/Transportation/TrafficSpaceUsageValue
A	Any use of the TrafficSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficSpaceUsageValue UML class as documented in the TrafficSpaceUsageValue section of the Transportation Data Dictionary .

Class TrafficSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations**Attributes**

12.15.46. Class TransportationSpaceClassValue

Subclass of <-- section,>>

Requirement 193	/req/Transportation/TransportationSpaceClassValue
A	Any use of the TransportationSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransportationSpaceClassValue UML class as documented in the TransportationSpaceClassValue section of the Transportation Data Dictionary .

Class TransportationSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.47. Class TransportationSpaceFunctionValue

Subclass of <-- section,>>

Requirement 194	/req/Transportation/TransportationSpaceFunctionValue
A	Any use of the TransportationSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransportationSpaceFunctionValue UML class as documented in the TransportationSpaceFunctionValue section of the Transportation Data Dictionary .

Class TransportationSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.48. Class TransportationSpaceUsageValue

Subclass of <-- section,>>

Requirement 195	/req/Transportation/TransportationSpaceUsageValue
-----------------	---

A	Any use of the TransportationSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TransportationSpaceUsageValue UML class as documented in the TransportationSpaceUsageValue section of the Transportation Data Dictionary .
---	--

Class TransportationSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.49. Class Waterway

Subclass of [AbstractTransportationSpace](#)

Requirement 196	/req/Transportation/Waterway
A	The Implementation Specification SHALL contain an element with the same definition as the Waterway UML class as documented in the Waterway section of the Transportation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Waterway UML class as documented in the Waterway section of the Transportation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Waterway UML class; including the name, definition, type, and cardinality of those documented in the Waterway section of the Transportation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Waterway UML class; including the name, definition, type, and cardinality of those documented in the Waterway section of the Transportation Data Dictionary .

Class Waterway

Definition:

Subtype Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Waterway</p> <p>Target Role: section</p> <p>Target Class: Section</p>
<p>Name:</p> <p>Type: Association</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Waterway</p> <p>Target Role: intersection</p> <p>Target Class: Intersection</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Waterway</p> <p>Target Role:</p> <p>Target Class: AbstractTransportationSpace</p>
Attributes
<p>Attribute Name: class</p> <p>Value Type: WaterwayClassValue</p> <p>Definition:</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: function</p> <p>Value Type: WaterwayFunctionValue</p> <p>Definition:</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: usage</p> <p>Value Type: WaterwayUsageValue</p> <p>Definition:</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>

12.15.50. Class WaterwayClassValue

Subclass of <← section,>>

Requirement 197	/req/Transportation/WaterwayClassValue
-----------------	--

A	Any use of the WaterwayClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterwayClassValue UML class as documented in the WaterwayClassValue section of the Transportation Data Dictionary .
---	---

Class WaterwayClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.51. Class WaterwayFunctionValue

Subclass of <-- section,>>

Requirement 198	/req/Transportation/WaterwayFunctionValue
A	Any use of the WaterwayFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterwayFunctionValue UML class as documented in the WaterwayFunctionValue section of the Transportation Data Dictionary .

Class WaterwayFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.52. Class WaterwayUsageValue

Subclass of <-- section,>>

Requirement 199	/req/Transportation/WaterwayUsageValue
A	Any use of the WaterwayUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WaterwayUsageValue UML class as documented in the WaterwayUsageValue section of the Transportation Data Dictionary .

Class WaterwayUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.15.53. Class GranularityValue

Subclass of <-- section,>>

Requirement 200	/req/Transportation/GranularityValue
A	Any use of the GranularityValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GranularityValue UML class as documented in the GranularityValue section of the Transportation Data Dictionary .

Class GranularityValue

Definition:

Subtype Of: <-- section,>>

Stereotype:

Associations

Attributes

Attribute Name: lane

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: way

Value Type:

Definition:

Multiplicity:

Stereotype:

12.15.54. Class TrafficDirectionValue

Subclass of <-- section,>>

Requirement 201	/req/Transportation/TrafficDirectionValue
-----------------	---

A	Any use of the TrafficDirectionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TrafficDirectionValue UML class as documented in the TrafficDirectionValue section of the Transportation Data Dictionary .
---	--

Class TrafficDirectionValue

Definition:

Subtype Of: <-- section,>>

Stereotype:

Associations

Attributes

Attribute Name: forwards

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: backwards

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: both

Value Type:

Definition:

Multiplicity:

Stereotype:

12.15.55. Additional Information

The following sections provide additional information which may not be readily available through the UML Model

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.16. Vegetation

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-vegetation>

Target type	Conceptual Model
Dependency	TBD

Dependency	TBD
------------	-----

TBD

The UML diagram of the Vegetation Model is depicted in [Vegetation UML Diagram](#). The Data Dictionary for the Vegetation Package is provided in section [Vegetation Model Data Dictionary](#).

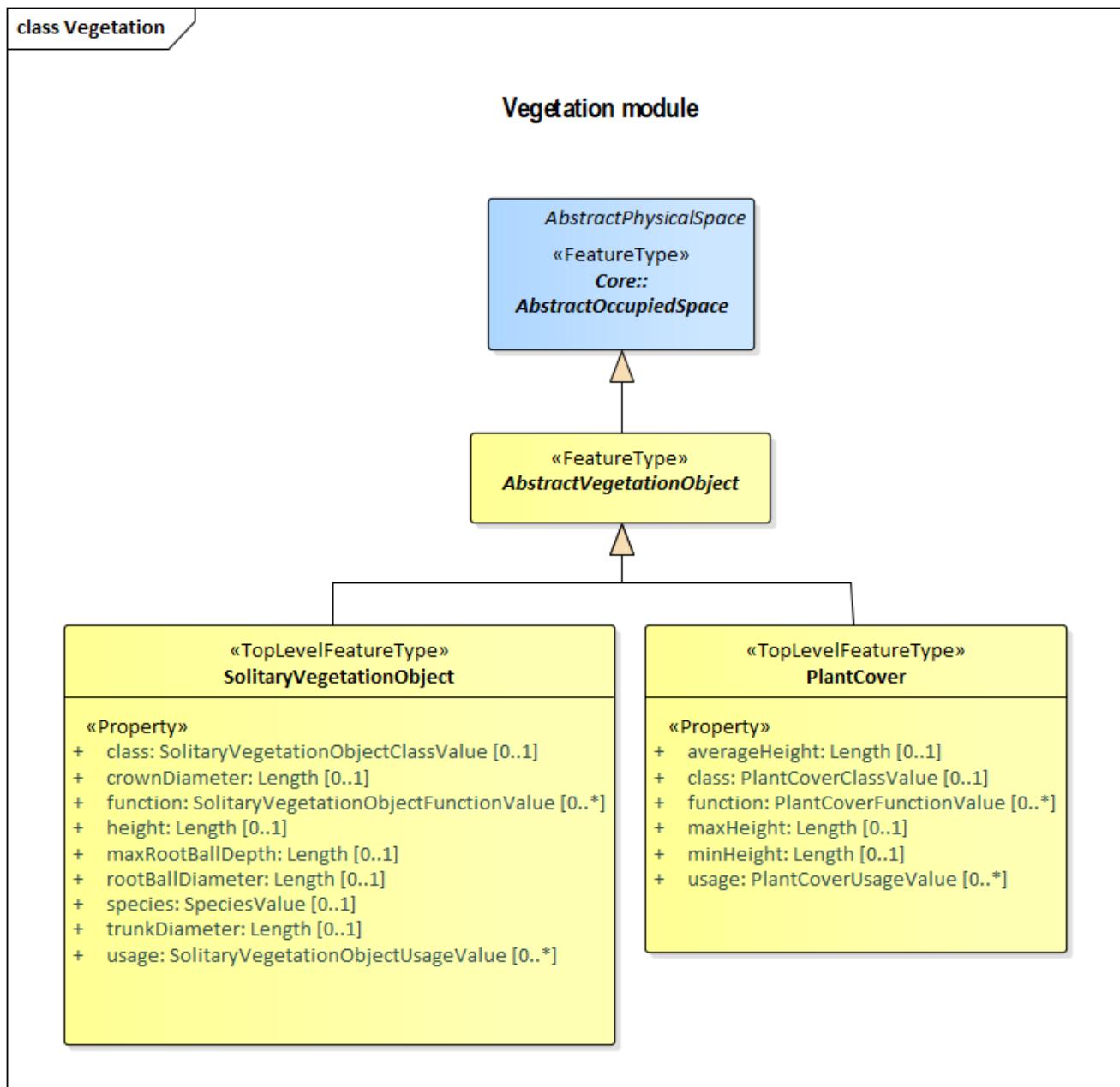


Figure 22. UML diagram of the Vegetation Model.

The [UML diagram of the Vegetation Model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.17. Vegetation Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.17.1. Class AbstractVegetationObject

Subclass of [AbstractOccupiedSpace](#)

Requirement 202	/req/Vegetation/AbstractVegetationObject
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractVegetationObject UML class as documented in the AbstractVegetationObject section of the Vegetation Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractVegetationObject UML class as documented in the AbstractVegetationObject section of the Vegetation Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractVegetationObject UML class; including the name, definition, type, and cardinality of those documented in the AbstractVegetationObject section of the Vegetation Data Dictionary .

Class AbstractVegetationObject

Definition:

Subtype Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractVegetationObject](#)

Target Role:

Target Class: [AbstractOccupiedSpace](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	PlantCover
Target Role:	
Target Class:	AbstractVegetationObject
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	SolitaryVegetationObject
Target Role:	
Target Class:	AbstractVegetationObject
Attributes	

12.17.2. Class PlantCover

Subclass of [AbstractVegetationObject](#)

Requirement 203	/req/Vegetation/PlantCover
A	The Implementation Specification SHALL contain an element with the same definition as the PlantCover UML class as documented in the PlantCover section of the Vegetation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the PlantCover UML class as documented in the PlantCover section of the Vegetation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the PlantCover UML class; including the name, definition, type, and cardinality of those documented in the PlantCover section of the Vegetation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the PlantCover UML class; including the name, definition, type, and cardinality of those documented in the PlantCover section of the Vegetation Data Dictionary .

Class PlantCover
Definition:
Subtype Of: AbstractVegetationObject
Stereotype: «TopLevelFeatureType»
Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	PlantCover
Target Role:	
Target Class:	AbstractVegetationObject
Attributes	
Attribute Name:	averageHeight
Value Type:	Length
Definition:	SIG3D: Average value of the heights of the grown-up plants
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	class
Value Type:	PlantCoverClassValue
Definition:	SIG3D: Classification of PlantCover as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	PlantCoverFunctionValue
Definition:	SIG3D: Function of PlantCover as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	maxHeight
Value Type:	Length
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	minHeight
Value Type:	Length
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	PlantCoverUsageValue
Definition:	SIG3D: Usage of PlantCover as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.17.3. Class PlantCoverClassValue

Subclass of <-- section,>>

Requirement 204	/req/Vegetation/PlantCoverClassValue
A	Any use of the PlantCoverClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the PlantCoverClassValue UML class as documented in the PlantCoverClassValue section of the Vegetation Data Dictionary .

Class PlantCoverClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.4. Class PlantCoverFunctionValue

Subclass of <-- section,>>

Requirement 205	/req/Vegetation/PlantCoverFunctionValue
A	Any use of the PlantCoverFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the PlantCoverFunctionValue UML class as documented in the PlantCoverFunctionValue section of the Vegetation Data Dictionary .

Class PlantCoverFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.5. Class PlantCoverUsageValue

Subclass of <-- section,>>

Requirement 206	/req/Vegetation/PlantCoverUsageValue
-----------------	--------------------------------------

A	Any use of the PlantCoverUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the PlantCoverUsageValue UML class as documented in the PlantCoverUsageValue section of the Vegetation Data Dictionary .
---	---

Class PlantCoverUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.6. Class SolitaryVegetationObject

Subclass of [AbstractVegetationObject](#)

Requirement 207	/req/Vegetation/SolitaryVegetationObject
A	The Implementation Specification SHALL contain an element with the same definition as the SolitaryVegetationObject UML class as documented in the SolitaryVegetationObject section of the Vegetation Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the SolitaryVegetationObject UML class as documented in the SolitaryVegetationObject section of the Vegetation Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the SolitaryVegetationObject UML class; including the name, definition, type, and cardinality of those documented in the SolitaryVegetationObject section of the Vegetation Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the SolitaryVegetationObject UML class; including the name, definition, type, and cardinality of those documented in the SolitaryVegetationObject section of the Vegetation Data Dictionary .

Class SolitaryVegetationObject

Definition:

Subtype Of: [AbstractVegetationObject](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	SolitaryVegetationObject
Target Role:	
Target Class:	AbstractVegetationObject
Attributes	
Attribute Name:	class
Value Type:	SolitaryVegetationObjectClassValue
Definition:	SIG3D: Classification of SolitaryVegetationObject as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	crownDiameter
Value Type:	Length
Definition:	SIG3D: Maximal diameter of the crown.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	SolitaryVegetationObjectFunctionValue
Definition:	SIG3D: Function of SolitaryVegetationObject as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	height
Value Type:	Length
Definition:	SIG3D: Distance between the highest point of the vegetation object and the lowest point of the terrain at the bottom of the object.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	maxRootBallDepth
Value Type:	Length
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	rootBallDiameter
Value Type:	Length
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	species
Value Type:	SpeciesValue
Definition:	SIG3D: Botanical classification of the SolitaryVegetationObject
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	trunkDiameter
Value Type:	Length
Definition:	SIG3D: Value of the trunk's diameter
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	SolitaryVegetationObjectUsageValue
Definition:	SIG3D: Usage of SolitaryVegetationObject as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.17.7. Class SolitaryVegetationObjectClassValue

Subclass of <-- section,>>

Requirement 208	/req/Vegetation/SolitaryVegetationObjectClassValue
A	Any use of the SolitaryVegetationObjectClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SolitaryVegetationObjectClassValue UML class as documented in the SolitaryVegetationObjectClassValue section of the Vegetation Data Dictionary .

Class SolitaryVegetationObjectClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.8. Class SolitaryVegetationObjectFunctionValue

Subclass of <-- section,>>

Requirement 209	/req/Vegetation/SolitaryVegetationObjectFunctionValue
-----------------	---

A	Any use of the SolitaryVegetationObjectFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SolitaryVegetationObjectFunctionValue UML class as documented in the SolitaryVegetationObjectFunctionValue section of the Vegetation Data Dictionary .
---	--

Class SolitaryVegetationObjectFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.9. Class SolitaryVegetationObjectUsageValue

Subclass of <-- section,>>

Requirement 210	/req/Vegetation/SolitaryVegetationObjectUsageValue
A	Any use of the SolitaryVegetationObjectUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SolitaryVegetationObjectUsageValue UML class as documented in the SolitaryVegetationObjectUsageValue section of the Vegetation Data Dictionary .

Class SolitaryVegetationObjectUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.10. Class SpeciesValue

Subclass of <-- section,>>

Requirement 211	/req/Vegetation/SpeciesValue
A	Any use of the SpeciesValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SpeciesValue UML class as documented in the SpeciesValue section of the Vegetation Data Dictionary .

Class SpeciesValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.17.11. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.18. City Furniture Model

TBD

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-cityfurniture>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

The UML diagram of the City Furniture model is depicted in the [City Furniture UML Diagram](#). The Data Dictionary for the City Furniture Package is provided in section [City Furniture Data Dictionary](#).

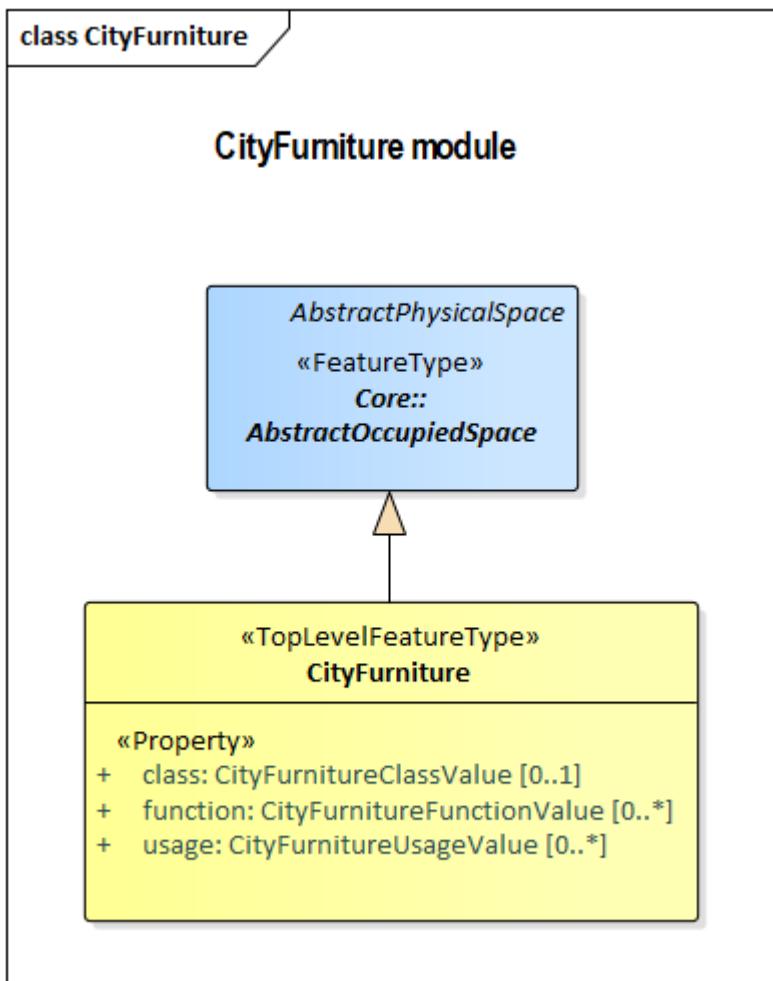


Figure 23. UML diagram of CityGML's city furniture model.

The [UML diagram of CityGML's city furniture model](#). is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.19. CityFurniture Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.19.1. Class CityFurniture

Subclass of [AbstractOccupiedSpace](#)

Requirement 212	/req/CityFurniture/CityFurniture
A	The Implementation Specification SHALL contain an element with the same definition as the CityFurniture UML class as documented in the CityFurniture section of the CityFurniture Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityFurniture UML class as documented in the CityFurniture section of the CityFurniture Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the CityFurniture UML class; including the name, definition, type, and cardinality of those documented in the CityFurniture section of the CityFurniture Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityFurniture UML class; including the name, definition, type, and cardinality of those documented in the CityFurniture section of the CityFurniture Data Dictionary .

Class CityFurniture

Definition:

Subtype Of: [AbstractOccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [CityFurniture](#)

Target Role:

Target Class: [AbstractOccupiedSpace](#)

Attributes

Attribute Name: class

Value Type: [CityFurnitureClassValue](#)

Definition: Classification of CityFurniture as given by the relevant national regulations, information communities, or specific applications. [SIG3D]

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: [CityFurnitureFunctionValue](#)

Definition: Specified function of CityFurniture as given by the relevant national regulations, information communities, or specific applications. [SIG3D]

Multiplicity: [0..*]

Stereotype: «Property»

Attribute Name:	usage
Value Type:	CityFurnitureUsageValue
Definition:	Actual usage of CityFurniture as given by the relevant national regulations, information communities, or specific applications. [SIG3D]
Multiplicity:	[0..*]
Stereotype:	«Property»

12.19.2. Class CityFurnitureClassValue

Subclass of <-- section,>>

Requirement 213	/req/CityFurniture/CityFurnitureClassValue
A	Any use of the CityFurnitureClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityFurnitureClassValue UML class as documented in the CityFurniture Data Dictionary .

Class CityFurnitureClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.19.3. Class CityFurnitureFunctionValue

Subclass of <-- section,>>

Requirement 214	/req/CityFurniture/CityFurnitureFunctionValue
A	Any use of the CityFurnitureFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityFurnitureFunctionValue UML class as documented in the CityFurniture Data Dictionary .

Class CityFurnitureFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.19.4. Class CityFurnitureUsageValue

Subclass of <-- section,>>

Requirement 215	/req/CityFurniture/CityFurnitureUsageValue
A	Any use of the CityFurnitureUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityFurnitureUsageValue UML class as documented in the CityFurnitureUsageValue section of the CityFurniture Data Dictionary .

Class CityFurnitureUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.19.5. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.20. Land Use

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-landuse>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Land Use Model is depicted in [Land Use UML Diagram](#). The Data Dictionary for the Land Use Package is provided in section [Land Use Data Dictionary](#).

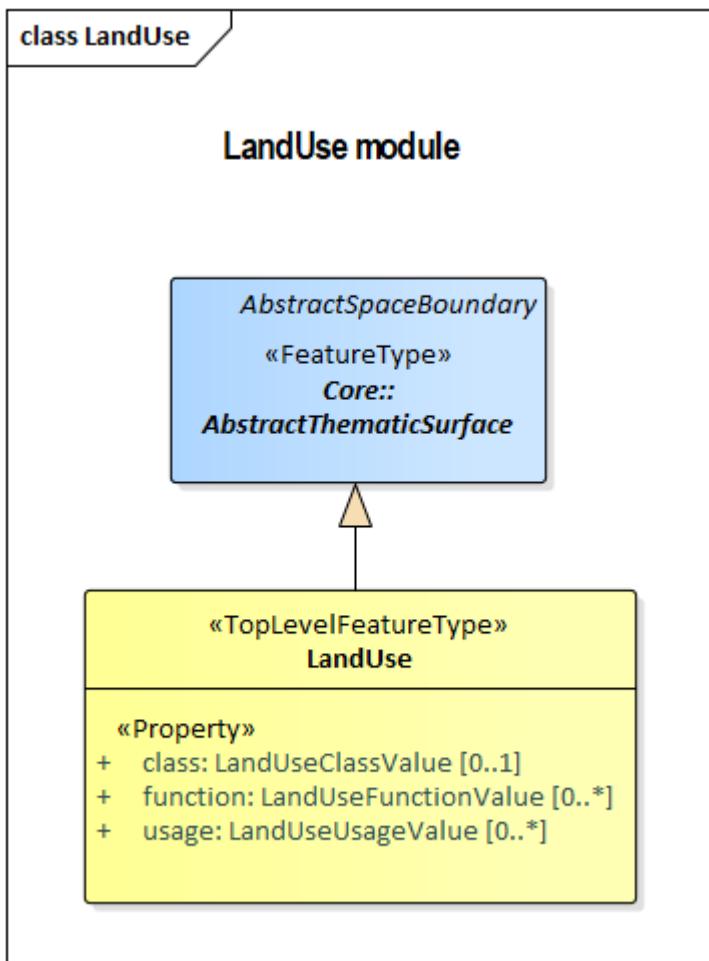


Figure 24. UML diagram of the Land Use Model.

The [UML diagram of the Land Use Model](#). is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.21. LandUse Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.21.1. Class LandUse

Subclass of [AbstractThematicSurface](#)

Requirement 216	/req/LandUse/LandUse
A	The Implementation Specification SHALL contain an element with the same definition as the LandUse UML class as documented in the LandUse section of the LandUse Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the LandUse UML class as documented in the LandUse section of the LandUse Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the LandUse UML class; including the name, definition, type, and cardinality of those documented in the LandUse section of the LandUse Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the LandUse UML class; including the name, definition, type, and cardinality of those documented in the LandUse section of the LandUse Data Dictionary .

Class LandUse

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [LandUse](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Attributes

Attribute Name: class

Value Type: LandUseClassValue

Definition: SIG3D: Classification of LandUse as given by the relevant national regulations, information communities or specific applications.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: LandUseFunctionValue

Definition: SIG3D: Specified function of LandUse as given by the relevant national regulations, information communities or specific applications.

Multiplicity: [0..*]

Stereotype: «Property»

Attribute Name:	usage
Value Type:	LandUseUsageValue
Definition:	SIG3D: Actual usage of LandUse as given by the relevant national regulations, information communities or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.21.2. Class LandUseClassValue

Subclass of <-- section,>>

Requirement 217	/req/LandUse/LandUseClassValue
A	Any use of the LandUseClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the LandUseClassValue UML class as documented in the LandUse Data Dictionary .

Class LandUseClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.21.3. Class LandUseFunctionValue

Subclass of <-- section,>>

Requirement 218	/req/LandUse/LandUseFunctionValue
A	Any use of the LandUseFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the LandUseFunctionValue UML class as documented in the LandUse Data Dictionary .

Class LandUseFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.21.4. Class LandUseUsageValue

Subclass of <-- section,>>

Requirement 219	/req/LandUse/LandUseUsageValue
A	Any use of the LandUseUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the LandUseUsageValue UML class as documented in the LandUseUsageValue section of the LandUse Data Dictionary .

Class LandUseUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.21.5. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.22. City Object Group

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-cityobjectgroup	
Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the City Object Group Model is depicted in [City Object Group UML Diagram](#). The Data Dictionary for the City Object Group Package is provided in section [City Object Group Data Dictionary](#).

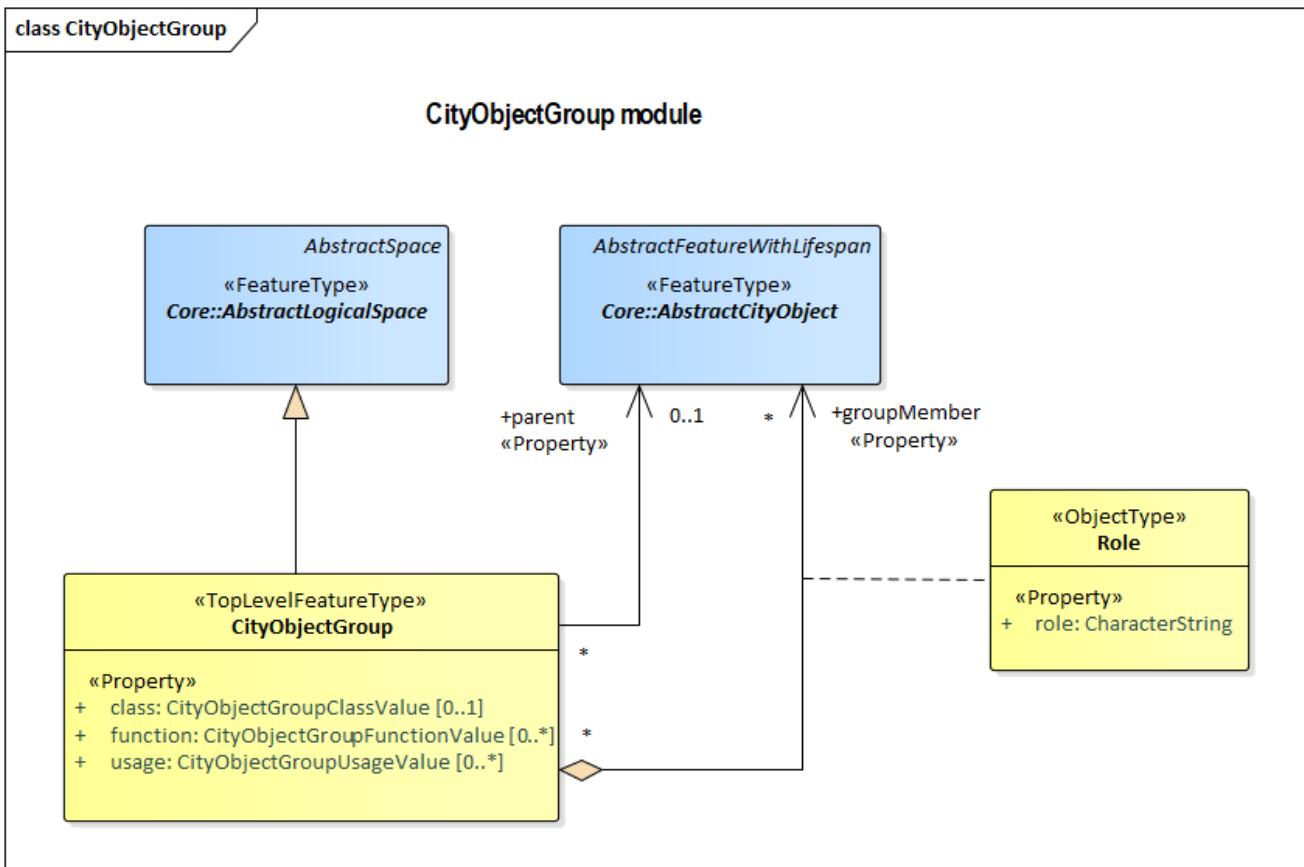


Figure 25. UML diagram of the City Object Group Model.

The [UML diagram of the City Object Group Model](#). is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.23. CityObjectGroup Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.23.1. Class CityObjectGroup

Subclass of [AbstractLogicalSpace](#)

Requirement 220	/req/CityObjectGroup/CityObjectGroup
A	The Implementation Specification SHALL contain an element with the same definition as the CityObjectGroup UML class as documented in the CityObjectGroup section of the CityObjectGroup Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CityObjectGroup UML class as documented in the CityObjectGroup section of the CityObjectGroup Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the CityObjectGroup UML class; including the name, definition, type, and cardinality of those documented in the CityObjectGroup section of the CityObjectGroup Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CityObjectGroup UML class; including the name, definition, type, and cardinality of those documented in the CityObjectGroup section of the CityObjectGroup Data Dictionary .

Class CityObjectGroup

Definition:

Subtype Of: [AbstractLogicalSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CityObjectGroup](#)

Target Role: parent

Target Class: [AbstractCityObject](#)

Name:

Type: AssociationClass

Direction: Source → Destination

Source Role:

Source Class: [CityObjectGroup](#)

Target Role: groupMember

Target Class: [AbstractCityObject](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [CityObjectGroup](#)

Target Role:

Target Class: [AbstractLogicalSpace](#)

Attributes

Attribute Name:	class
Value Type:	CityObjectGroupClassValue
Definition:	SIG3D: General semantical meaning of the aggregation. Classification of the aggregation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	CityObjectGroupFunctionValue
Definition:	SIG3D: Specific semantic meaning of the aggregation. Function of the aggregation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	CityObjectGroupUsageValue
Definition:	SIG3D: Usage of the aggregation as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

12.23.2. Class CityObjectGroupClassValue

Subclass of <-- section,>>

Requirement 221	/req/CityObjectGroup/CityObjectGroupClassValue
A	Any use of the CityObjectGroupClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityObjectGroupClassValue UML class as documented in the CityObjectGroupClassValue section of the CityObjectGroup Data Dictionary .

Class CityObjectGroupClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.23.3. Class CityObjectGroupFunctionValue

Subclass of <-- section,>>

Requirement 222	/req/CityObjectGroup/CityObjectGroupFunctionValue
-----------------	---

A	Any use of the CityObjectGroupFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityObjectGroupFunctionValue UML class as documented in the CityObjectGroupFunctionValue section of the CityObjectGroup Data Dictionary .
---	--

Class CityObjectGroupFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.23.4. Class CityObjectGroupUsageValue

Subclass of <-- section,>>

Requirement 223	/req/CityObjectGroup/CityObjectGroupUsageValue
A	Any use of the CityObjectGroupUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the CityObjectGroupUsageValue UML class as documented in the CityObjectGroupUsageValue section of the CityObjectGroup Data Dictionary .

Class CityObjectGroupUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.23.5. Class Role

Subclass of <-- section,>>

Requirement 224	/req/CityObjectGroup/Role
A	The Implementation Specification SHALL contain an element with the same definition as the Role UML class as documented in the Role section of the CityObjectGroup Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Role UML class as documented in the Role section of the CityObjectGroup Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Role UML class; including the name, definition, type, and cardinality of those documented in the Role section of the CityObjectGroup Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Role UML class; including the name, definition, type, and cardinality of those documented in the Role section of the CityObjectGroup Data Dictionary .

Class Role
Definition:
Subtype Of: <-> section,>>
Stereotype: «ObjectType»
Associations
Attributes
Attribute Name: role
Value Type: CharacterString
Definition: SIG3D: Description of the role.
Multiplicity:
Stereotype: «Property»

12.23.6. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.24. Generics

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-generics	
Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Generics Model is depicted in [Generics UML Diagram](#). The Data Dictionary for the Generics Package is provided in section [Generics Data Dictionary](#).

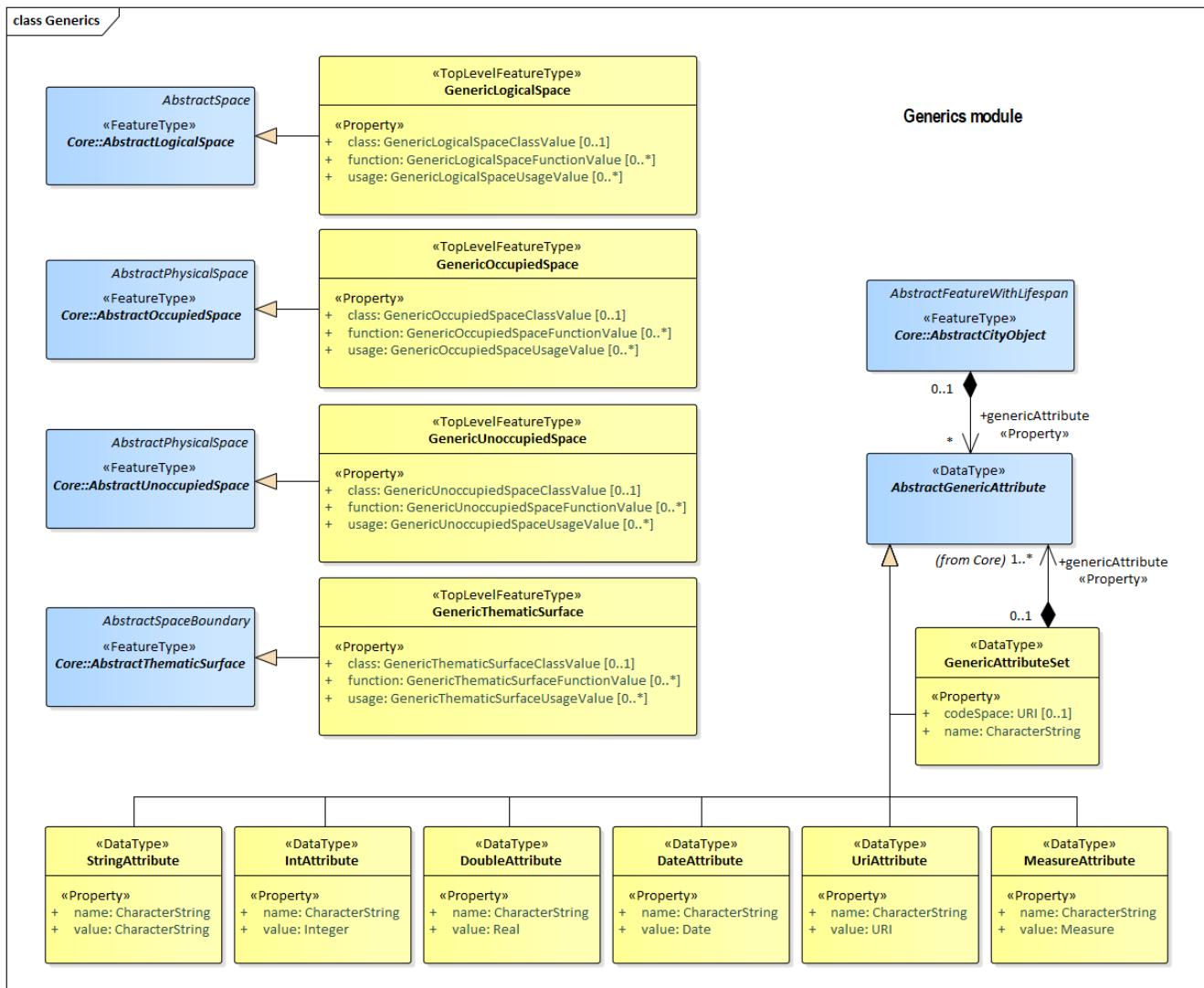


Figure 26. UML diagram of the Generics Model.

The [UML diagram of the Generics Model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.25. Generics Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.25.1. Class GenericLogicalSpace

Subclass of [AbstractLogicalSpace](#)

Requirement 225	/req/Generics/GenericLogicalSpace
A	The Implementation Specification SHALL contain an element with the same definition as the GenericLogicalSpace UML class as documented in the GenericLogicalSpace section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericLogicalSpace UML class as documented in the GenericLogicalSpace section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GenericLogicalSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericLogicalSpace section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericLogicalSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericLogicalSpace section of the Generics Data Dictionary .

Class GenericLogicalSpace

Definition:

Subtype Of: [AbstractLogicalSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [GenericLogicalSpace](#)

Target Role:

Target Class: [AbstractLogicalSpace](#)

Attributes

Attribute Name: class

Value Type: GenericLogicalSpaceClassValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: GenericLogicalSpaceFunctionValue

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

Attribute Name:	usage
Value Type:	GenericLogicalSpaceUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.25.2. Class GenericLogicalSpaceClassValue

Subclass of <-- section,>>

Requirement 226	/req/Generics/GenericLogicalSpaceClassValue
A	Any use of the GenericLogicalSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericLogicalSpaceClassValue UML class as documented in the GenericLogicalSpaceClassValue section of the Generics Data Dictionary .

Class GenericLogicalSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.25.3. Class GenericLogicalSpaceFunctionValue

Subclass of <-- section,>>

Requirement 227	/req/Generics/GenericLogicalSpaceFunctionValue
A	Any use of the GenericLogicalSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericLogicalSpaceFunctionValue UML class as documented in the GenericLogicalSpaceFunctionValue section of the Generics Data Dictionary .

Class GenericLogicalSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.25.4. Class GenericLogicalSpaceUsageValue

Subclass of <-- section,>>

Requirement 228	/req/Generics/GenericLogicalSpaceUsageValue
A	Any use of the GenericLogicalSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericLogicalSpaceUsageValue UML class as documented in the GenericLogicalSpaceUsageValue section of the Generics Data Dictionary .

Class GenericLogicalSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.25.5. Class GenericOccupiedSpace

Subclass of [AbstractOccupiedSpace](#)

Requirement 229	/req/Generics/GenericOccupiedSpace
A	The Implementation Specification SHALL contain an element with the same definition as the GenericOccupiedSpace UML class as documented in the GenericOccupiedSpace section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericOccupiedSpace UML class as documented in the GenericOccupiedSpace section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GenericOccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericOccupiedSpace section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericOccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericOccupiedSpace section of the Generics Data Dictionary .

Class GenericOccupiedSpace

Definition:	
Subtype Of:	AbstractOccupiedSpace
Stereotype:	«TopLevelFeatureType»
Associations	
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GenericOccupiedSpace
Target Role:	
Target Class:	AbstractOccupiedSpace
Attributes	
Attribute Name:	
Value Type:	GenericOccupiedSpaceClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	
Value Type:	GenericOccupiedSpaceFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	
Value Type:	GenericOccupiedSpaceUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.25.6. Class GenericOccupiedSpaceClassValue

Subclass of <– section,>>

Requirement 230	/req/Generics/GenericOccupiedSpaceClassValue
A	Any use of the GenericOccupiedSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericOccupiedSpaceClassValue UML class as documented in the GenericOccupiedSpaceClassValue section of the Generics Data Dictionary .

Class GenericOccupiedSpaceClassValue	
Definition:	
Subtype Of:	<– section,>>
Stereotype:	«CodeList»

Associations**Attributes**

12.25.7. Class GenericOccupiedSpaceFunctionValue

Subclass of <-- section,>>

Requirement 231	/req/Generics/GenericOccupiedSpaceFunctionValue
A	Any use of the GenericOccupiedSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericOccupiedSpaceFunctionValue UML class as documented in the GenericOccupiedSpaceFunctionValue section of the Generics Data Dictionary .

Class GenericOccupiedSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations**Attributes**

12.25.8. Class GenericOccupiedSpaceUsageValue

Subclass of <-- section,>>

Requirement 232	/req/Generics/GenericOccupiedSpaceUsageValue
A	Any use of the GenericOccupiedSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericOccupiedSpaceUsageValue UML class as documented in the GenericOccupiedSpaceUsageValue section of the Generics Data Dictionary .

Class GenericOccupiedSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations**Attributes**

12.25.9. Class GenericThematicSurface

Subclass of [AbstractThematicSurface](#)

Requirement 233	/req/Generics/GenericThematicSurface
A	The Implementation Specification SHALL contain an element with the same definition as the GenericThematicSurface UML class as documented in the GenericThematicSurface section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericThematicSurface UML class as documented in the GenericThematicSurface section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GenericThematicSurface UML class; including the name, definition, type, and cardinality of those documented in the GenericThematicSurface section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericThematicSurface UML class; including the name, definition, type, and cardinality of those documented in the GenericThematicSurface section of the Generics Data Dictionary .

Class GenericThematicSurface

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [GenericThematicSurface](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Attributes

Attribute Name: class

Value Type: [GenericThematicSurfaceClassValue](#)

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	function
Value Type:	GenericThematicSurfaceFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	GenericThematicSurfaceUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.25.10. Class GenericThematicSurfaceClassValue

Subclass of <-- section,>>

Requirement 234	/req/Generics/GenericThematicSurfaceClassValue
A	Any use of the GenericThematicSurfaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericThematicSurfaceClassValue UML class as documented in the GenericThematicSurfaceClassValue section of the Generics Data Dictionary .

Class GenericThematicSurfaceClassValue

Definition:
 Subtype Of: <-- section,>>
 Stereotype: «CodeList»

Associations

Attributes

12.25.11. Class GenericThematicSurfaceFunctionValue

Subclass of <-- section,>>

Requirement 235	/req/Generics/GenericThematicSurfaceFunctionValue
A	Any use of the GenericThematicSurfaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericThematicSurfaceFunctionValue UML class as documented in the GenericThematicSurfaceFunctionValue section of the Generics Data Dictionary .

Class GenericThematicSurfaceFunctionValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.25.12. Class GenericThematicSurfaceUsageValue

Subclass of <-- section,>>

Requirement 236	/req/Generics/GenericThematicSurfaceUsageValue
A	Any use of the GenericThematicSurfaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericThematicSurfaceUsageValue UML class as documented in the GenericThematicSurfaceUsageValue section of the Generics Data Dictionary .

Class GenericThematicSurfaceUsageValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.25.13. Class GenericUnoccupiedSpace

Subclass of [AbstractUnoccupiedSpace](#)

Requirement 237	/req/Generics/GenericUnoccupiedSpace
A	The Implementation Specification SHALL contain an element with the same definition as the GenericUnoccupiedSpace UML class as documented in the GenericUnoccupiedSpace section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericUnoccupiedSpace UML class as documented in the GenericUnoccupiedSpace section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GenericUnoccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericUnoccupiedSpace section of the Generics Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericUnoccupiedSpace UML class; including the name, definition, type, and cardinality of those documented in the GenericUnoccupiedSpace section of the Generics Data Dictionary .
---	--

Class GenericUnoccupiedSpace

Definition:

Subtype Of: [AbstractUnoccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [GenericUnoccupiedSpace](#)

Target Role:

Target Class: [AbstractUnoccupiedSpace](#)

Attributes

Attribute Name: class

Value Type: GenericUnoccupiedSpaceClassValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: GenericUnoccupiedSpaceFunctionValue

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

Attribute Name: usage

Value Type: GenericUnoccupiedSpaceUsageValue

Definition:

Multiplicity: [0..*]

Stereotype: «Property»

12.25.14. Class GenericUnoccupiedSpaceClassValue

Subclass of <← section,>>

Requirement 238	/req/Generics/GenericUnoccupiedSpaceClassValue
-----------------	--

A	Any use of the GenericUnoccupiedSpaceClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericUnoccupiedSpaceClassValue UML class as documented in the GenericUnoccupiedSpaceClassValue section of the Generics Data Dictionary .
---	---

Class GenericUnoccupiedSpaceClassValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.25.15. Class GenericUnoccupiedSpaceFunctionValue

Subclass of <-- section,>>

Requirement 239	/req/Generics/GenericUnoccupiedSpaceFunctionValue
A	Any use of the GenericUnoccupiedSpaceFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericUnoccupiedSpaceFunctionValue UML class as documented in the GenericUnoccupiedSpaceFunctionValue section of the Generics Data Dictionary .

Class GenericUnoccupiedSpaceFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.25.16. Class GenericUnoccupiedSpaceUsageValue

Subclass of <-- section,>>

Requirement 240	/req/Generics/GenericUnoccupiedSpaceUsageValue
A	Any use of the GenericUnoccupiedSpaceUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the GenericUnoccupiedSpaceUsageValue UML class as documented in the GenericUnoccupiedSpaceUsageValue section of the Generics Data Dictionary .

Class GenericUnoccupiedSpaceUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.25.17. Class DateAttribute

Subclass of [AbstractGenericAttribute](#)

Requirement 241	/req/Generics/ DateAttribute
A	Any use of the DateAttribute type in the Implementation Specification SHALL have the same definition as the DateAttribute UML class as documented in the DateAttribute section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DateAttribute UML class as documented in the DateAttribute section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the DateAttribute UML class; including the name, definition, type, and cardinality of those documented in the DateAttribute section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DateAttribute UML class; including the name, definition, type, and cardinality of those documented in the DateAttribute section of the Generics Data Dictionary .

Class DateAttribute

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [DateAttribute](#)

Target Role:

Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: name
Value Type: CharacterString
Definition:
Multiplicity:
Stereotype: «Property»

Attribute Name: value
Value Type: Date
Definition: SIG3D: Value of the Generic Attribute.
Multiplicity:
Stereotype: «Property»

12.25.18. Class DoubleAttribute

Subclass of [AbstractGenericAttribute](#)

Requirement 242	/req/Generics/DoubleAttribute
A	Any use of the DoubleAttribute type in the Implementation Specification SHALL have the same definition as the DoubleAttribute UML class as documented in the DoubleAttribute section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoubleAttribute UML class as documented in the DoubleAttribute section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the DoubleAttribute UML class; including the name, definition, type, and cardinality of those documented in the DoubleAttribute section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoubleAttribute UML class; including the name, definition, type, and cardinality of those documented in the DoubleAttribute section of the Generics Data Dictionary .

Class DoubleAttribute

Definition:
Subtype Of: <-> section
Stereotype: «DataType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	DoubleAttribute
Target Role:	
Target Class:	AbstractGenericAttribute

Attributes

Attribute Name:	name
Value Type:	CharacterString
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	value
Value Type:	Real
Definition:	SIG3D: Value of the Generic Attribute.
Multiplicity:	
Stereotype:	«Property»

12.25.19. Class GenericAttributeSet

Subclass of [AbstractGenericAttribute](#)

Requirement 243	/req/Generics/GenericAttributeSet
A	Any use of the GenericAttributeSet type in the Implementation Specification SHALL have the same definition as the GenericAttributeSet UML class as documented in the GenericAttributeSet section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericAttributeSet UML class as documented in the GenericAttributeSet section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GenericAttributeSet UML class; including the name, definition, type, and cardinality of those documented in the GenericAttributeSet section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericAttributeSet UML class; including the name, definition, type, and cardinality of those documented in the GenericAttributeSet section of the Generics Data Dictionary .

Class GenericAttributeSet

Definition:
Subtype Of: <-- section,>>
Stereotype: «DataType»

Associations

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [GenericAttributeSet](#)
Target Role: genericAttribute
Target Class: [AbstractGenericAttribute](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [GenericAttributeSet](#)
Target Role:
Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: codeSpace
Value Type: URI
Definition: SIG3D: Codespace idcentifier of the Generic AttributeSet.
Multiplicity: [0..1]
Stereotype: «Property»

Attribute Name: name
Value Type: CharacterString
Definition:
Multiplicity:
Stereotype: «Property»

12.25.20. Class IntAttribute

Subclass of [AbstractGenericAttribute](#)

Requirement 244	/req/Generics/IntAttribute
A	Any use of the IntAttribute type in the Implementation Specification SHALL have the same definition as the IntAttribute UML class as documented in the IntAttribute section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the IntAttribute UML class as documented in the IntAttribute section of the Generics Data Dictionary .

C	The implementation Specification SHALL represent the attributes of the IntAttribute UML class; including the name, definition, type, and cardinality of those documented in the IntAttribute section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the IntAttribute UML class; including the name, definition, type, and cardinality of those documented in the IntAttribute section of the Generics Data Dictionary .

Class IntAttribute

Definition:

Subtype Of: <-> section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [IntAttribute](#)

Target Role:

Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: name

Value Type: CharacterString

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Integer

Definition: SIG3D: Value of the Generic Attribute.

Multiplicity:

Stereotype: «Property»

12.25.21. Class MeasureAttribute

Subclass of [AbstractGenericAttribute](#)

Requirement 245	/req/Generics/MeasureAttribute
A	Any use of the MeasureAttribute type in the Implementation Specification SHALL have the same definition as the MeasureAttribute UML class as documented in the MeasureAttribute section of the Generics Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the MeasureAttribute UML class as documented in the MeasureAttribute section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the MeasureAttribute UML class; including the name, definition, type, and cardinality of those documented in the MeasureAttribute section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the MeasureAttribute UML class; including the name, definition, type, and cardinality of those documented in the MeasureAttribute section of the Generics Data Dictionary .

Class MeasureAttribute

Definition:

Subtype Of: <->section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [MeasureAttribute](#)

Target Role:

Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: name

Value Type: CharacterString

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: Measure

Definition: SIG3D: Value of the Generic Attribute.

Multiplicity:

Stereotype: «Property»

12.25.22. Class StringAttribute

Subclass of [AbstractGenericAttribute](#)

Requirement 246	/req/Generics/StringAttribute
-----------------	-------------------------------

A	Any use of the StringAttribute type in the Implementation Specification SHALL have the same definition as the StringAttribute UML class as documented in the StringAttribute section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the StringAttribute UML class as documented in the StringAttribute section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the StringAttribute UML class; including the name, definition, type, and cardinality of those documented in the StringAttribute section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the StringAttribute UML class; including the name, definition, type, and cardinality of those documented in the StringAttribute section of the Generics Data Dictionary .

Class StringAttribute

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [StringAttribute](#)

Target Role:

Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: name

Value Type: CharacterString

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: value

Value Type: CharacterString

Definition: SIG3D: Value of the Generic Attribute.

Multiplicity:

Stereotype: «Property»

12.25.23. Class UriAttribute

Subclass of [AbstractGenericAttribute](#)

Requirement 247	/req/Generics/UriAttribute
A	Any use of the UriAttribute type in the Implementation Specification SHALL have the same definition as the UriAttribute UML class as documented in the UriAttribute section of the Generics Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the UriAttribute UML class as documented in the UriAttribute section of the Generics Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the UriAttribute UML class; including the name, definition, type, and cardinality of those documented in the UriAttribute section of the Generics Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the UriAttribute UML class; including the name, definition, type, and cardinality of those documented in the UriAttribute section of the Generics Data Dictionary .

Class UriAttribute

Definition:

Subtype Of: <-> section,>>

Stereotype: «DataType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [UriAttribute](#)

Target Role:

Target Class: [AbstractGenericAttribute](#)

Attributes

Attribute Name: name

Value Type: CharacterString

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name:	value
Value Type:	URI
Definition:	SIG3D: Value of the Generic Attribute.
Multiplicity:	
Stereotype:	«Property»

12.25.24. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.26. Construction

Requirements Class	
http://www.opengis.net/spec/CityGML/3.1/req/req-class-construction	
Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Construction Model is depicted in [Construction UML Diagram](#). The Data Dictionary for the Construction Package is provided in section [Construction Data Dictionary](#).

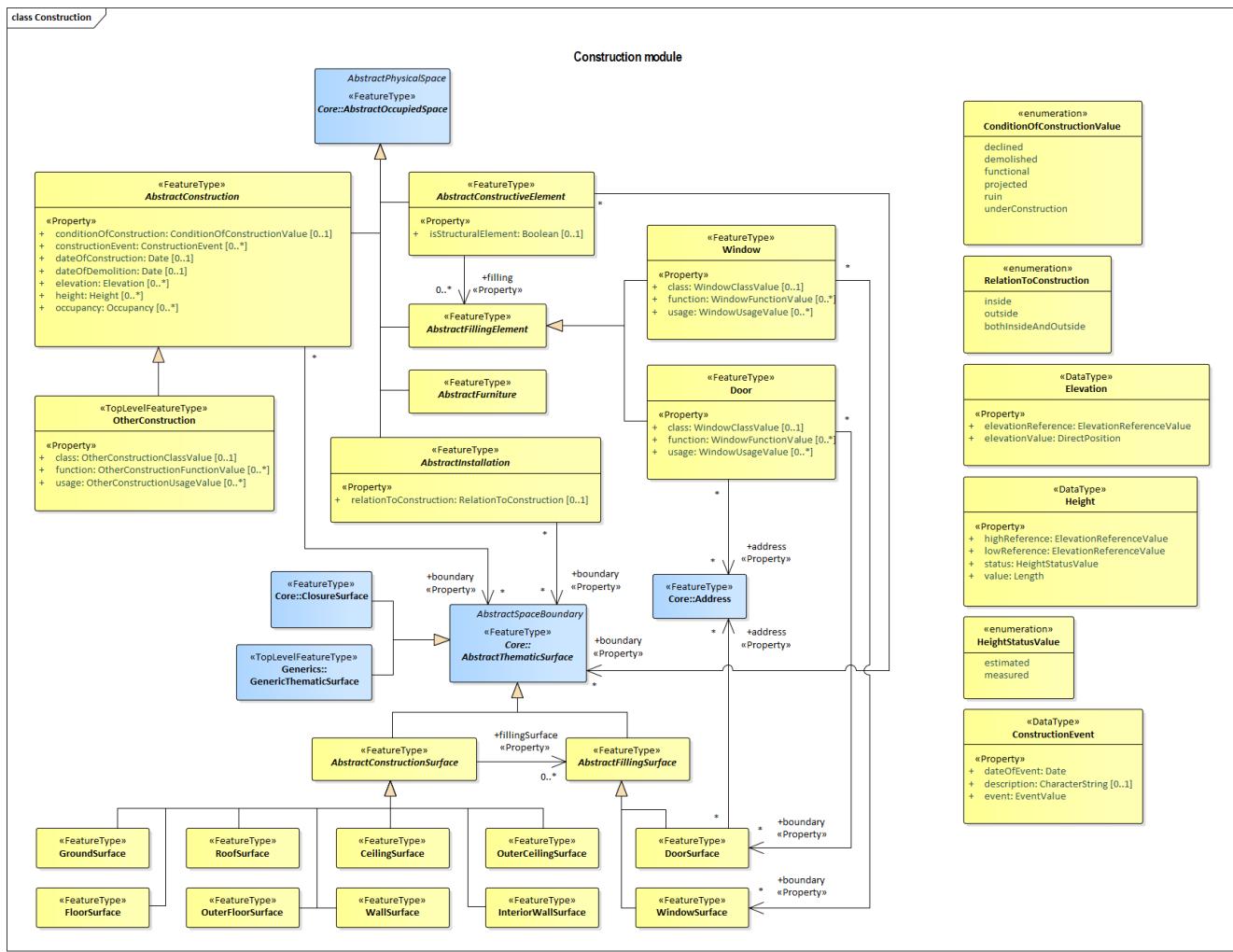


Figure 27. UML diagram of the Construction Model.

The [UML diagram of the Construction Model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.27. Construction Model Data Dictionary

Description:

Stereotype: `«ApplicationSchema»`

Parent Package: CityGML

12.27.1. Class AbstractConstruction

Subclass of [AbstractOccupiedSpace](#)

Requirement 248	/req/Construction/AbstractConstruction
-----------------	--

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractConstruction UML class as documented in the AbstractConstruction section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractConstruction UML class as documented in the AbstractConstruction section of the Construction Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractConstruction UML class; including the name, definition, type, and cardinality of those documented in the AbstractConstruction section of the Construction Data Dictionary .

Class AbstractConstruction

Definition:

Subtype Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstruction](#)

Target Role:

Target Class: [AbstractOccupiedSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstruction](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [OtherConstruction](#)

Target Role:

Target Class: [AbstractConstruction](#)

<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractTunnel</p> <p>Target Role:</p> <p>Target Class: AbstractConstruction</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBuilding</p> <p>Target Role:</p> <p>Target Class: AbstractConstruction</p>
<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: AbstractBridge</p> <p>Target Role:</p> <p>Target Class: AbstractConstruction</p>
<p>Attributes</p> <p>Attribute Name: conditionOfConstruction</p> <p>Value Type: ConditionOfConstructionValue</p> <p>Definition: [INSPIRE] Status of the construction.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: constructionEvent</p> <p>Value Type: ConstructionEvent</p> <p>Definition: Date of renovation of a construction.</p> <p>Multiplicity: [0..*]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: dateOfConstruction</p> <p>Value Type: Date</p> <p>Definition: Date of completion of a construction.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: dateOfDemolition</p> <p>Value Type: Date</p> <p>Definition: Date of demolition of a construction.</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

Attribute Name:	elevation
Value Type:	Elevation
Definition:	[INSPIRE] Vertically-constrained dimensional property consisting of an absolute measure referenced to a well-defined surface which is commonly taken as origin (geoid, water level, etc.).
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	height
Value Type:	Height
Definition:	[INSPIRE] Height above ground.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.27.2. Class AbstractConstructionSurface

Subclass of [AbstractThematicSurface](#)

Requirement 249	/req/Construction/AbstractConstructionSurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractConstructionSurface UML class as documented in the AbstractConstructionSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractConstructionSurface UML class as documented in the AbstractConstructionSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractConstructionSurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractConstructionSurface section of the Construction Data Dictionary .

Class AbstractConstructionSurface

Definition:
Subtype Of: [AbstractThematicSurface](#)
StereoType: «FeatureType»

Associations

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [AbstractConstructionSurface](#)
Target Role:
Target Class: [AbstractThematicSurface](#)

Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: [AbstractConstructionSurface](#)
Target Role: fillingSurface
Target Class: [AbstractFillingSurface](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [FloorSurface](#)
Target Role:
Target Class: [AbstractConstructionSurface](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [OuterFloorSurface](#)
Target Role:
Target Class: [AbstractConstructionSurface](#)

Name:
Type: Generalization
Direction: Source → Destination
Source Role:
Source Class: [CeilingSurface](#)
Target Role:
Target Class: [AbstractConstructionSurface](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	OuterCeilingSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	RoofSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GroundSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	InteriorWallSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	WallSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Attributes	

12.27.3. Class AbstractConstructiveElement

Subclass of [AbstractOccupiedSpace](#)

Requirement 250	/req/Construction/AbstractConstructiveElement
------------------------	---

A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractConstructiveElement UML class as documented in the AbstractConstructiveElement section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractConstructiveElement UML class as documented in the AbstractConstructiveElement section of the Construction Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractConstructiveElement UML class; including the name, definition, type, and cardinality of those documented in the AbstractConstructiveElement section of the Construction Data Dictionary .

Class AbstractConstructiveElement

Definition:

Subtype Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstructiveElement](#)

Target Role: filling

Target Class: [AbstractFillingElement](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstructiveElement](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstructiveElement](#)

Target Role:

Target Class: [AbstractOccupiedSpace](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	BuildingConstructiveElement
Target Role:	
Target Class:	AbstractConstructiveElement
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	BridgeConstructiveElement
Target Role:	
Target Class:	AbstractConstructiveElement
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TunnelConstructiveElement
Target Role:	
Target Class:	AbstractConstructiveElement
Attributes	
Attribute Name:	<code>isStructuralElement</code>
Value Type:	Boolean
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.27.4. Class AbstractFillingElement

Subclass of [AbstractOccupiedSpace](#)

Requirement 251	/req/Construction/AbstractFillingElement
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFillingElement UML class as documented in the AbstractFillingElement section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFillingElement UML class as documented in the AbstractFillingElement section of the Construction Data Dictionary .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFillingElement UML class; including the name, definition, type, and cardinality of those documented in the AbstractFillingElement section of the Construction Data Dictionary .
---	--

Class AbstractFillingElement

Definition:

Subtype Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractFillingElement](#)

Target Role:

Target Class: [AbstractOccupiedSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstructiveElement](#)

Target Role: filling

Target Class: [AbstractFillingElement](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Window](#)

Target Role:

Target Class: [AbstractFillingElement](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Door](#)

Target Role:

Target Class: [AbstractFillingElement](#)

Attributes

12.27.5. Class AbstractFillingSurface

Subclass of [AbstractThematicSurface](#)

Requirement 252	/req/Construction/AbstractFillingSurface
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFillingSurface UML class as documented in the AbstractFillingSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFillingSurface UML class as documented in the AbstractFillingSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFillingSurface UML class; including the name, definition, type, and cardinality of those documented in the AbstractFillingSurface section of the Construction Data Dictionary .

Class AbstractFillingSurface

Definition:

Subtype Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractFillingSurface](#)

Target Role:

Target Class: [AbstractThematicSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractConstructionSurface](#)

Target Role: fillingSurface

Target Class: [AbstractFillingSurface](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	WindowSurface
Target Role:	
Target Class:	AbstractFillingSurface
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	DoorSurface
Target Role:	
Target Class:	AbstractFillingSurface
Attributes	

12.27.6. Class AbstractFurniture

Subclass of [AbstractOccupiedSpace](#)

Requirement 253	/req/Construction/AbstractFurniture
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractFurniture UML class as documented in the AbstractFurniture section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractFurniture UML class as documented in the AbstractFurniture section of the Construction Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractFurniture UML class; including the name, definition, type, and cardinality of those documented in the AbstractFurniture section of the Construction Data Dictionary .

Class AbstractFurniture
Definition:
Subtype Of: AbstractOccupiedSpace
Stereotype: «FeatureType»
Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	AbstractFurniture
Target Role:	
Target Class:	AbstractOccupiedSpace
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	BridgeFurniture
Target Role:	
Target Class:	AbstractFurniture
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TunnelFurniture
Target Role:	
Target Class:	AbstractFurniture
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	BuildingFurniture
Target Role:	
Target Class:	AbstractFurniture
Attributes	

12.27.7. Class AbstractInstallation

Subclass of [AbstractOccupiedSpace](#)

Requirement 254	/req/Construction/AbstractInstallation
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractInstallation UML class as documented in the AbstractInstallation section of the Construction Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractInstallation UML class as documented in the AbstractInstallation section of the Construction Data Dictionary .

C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractInstallation UML class; including the name, definition, type, and cardinality of those documented in the AbstractInstallation section of the Construction Data Dictionary .
---	--

Class AbstractInstallation

Definition:

Subtype Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractInstallation](#)

Target Role:

Target Class: [AbstractOccupiedSpace](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [AbstractInstallation](#)

Target Role: boundary

Target Class: [AbstractThematicSurface](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [BridgeInstallation](#)

Target Role:

Target Class: [AbstractInstallation](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [TunnelInstallation](#)

Target Role:

Target Class: [AbstractInstallation](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	BuildingInstallation
Target Role:	
Target Class:	AbstractInstallation
Attributes	
Attribute Name:	relationToConstruction
Value Type:	RelationToConstruction
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.27.8. Class CeilingSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 255	/req/Construction/CeilingSurface
A	The Implementation Specification SHALL contain an element with the same definition as the CeilingSurface UML class as documented in the CeilingSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CeilingSurface UML class as documented in the CeilingSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the CeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the CeilingSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the CeilingSurface section of the Construction Data Dictionary .

Class CeilingSurface
Definition:
Subtype Of: AbstractConstructionSurface
Stereotype: «FeatureType»
Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	CeilingSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Attributes	

12.27.9. Class Door

Subclass of [AbstractFillingElement](#)

Requirement 256	/req/Construction/Door
A	The Implementation Specification SHALL contain an element with the same definition as the Door UML class as documented in the Door section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Door UML class as documented in the Door section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Door UML class; including the name, definition, type, and cardinality of those documented in the Door section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Door UML class; including the name, definition, type, and cardinality of those documented in the Door section of the Construction Data Dictionary .

Class Door
Definition:
Subtype Of: AbstractFillingElement
Stereotype: «FeatureType»
Associations
Name:
Type: Association
Direction: Source → Destination
Source Role:
Source Class: Door
Target Role: boundary
Target Class: DoorSurface

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Door
Target Role:	
Target Class:	AbstractFillingElement
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Door
Target Role:	address
Target Class:	Address
Attributes	
Attribute Name:	class
Value Type:	WindowClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	WindowFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	WindowUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.27.10. Class DoorClassValue

Subclass of <← section,>>

Requirement 257	/req/Construction/DoorClassValue
A	Any use of the DoorClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorClassValue UML class as documented in the DoorClassValue section of the Construction Data Dictionary .

Class DoorClassValue

Definition:
Subtype Of: <--section,>>
StereoType: «CodeList»

Associations

Attributes

12.27.11. Class DoorFunctionValue

Subclass of <--section,>>

Requirement 258	/req/Construction/DoorFunctionValue
A	Any use of the DoorFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorFunctionValue UML class as documented in the DoorFunctionValue section of the Construction Data Dictionary .

Class DoorFunctionValue

Definition:
Subtype Of: <--section,>>
StereoType: «CodeList»

Associations

Attributes

12.27.12. Class DoorSurface

Subclass of [AbstractFillingSurface](#)

Requirement 259	/req/Construction/DoorSurface
A	The Implementation Specification SHALL contain an element with the same definition as the DoorSurface UML class as documented in the DoorSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the DoorSurface UML class as documented in the DoorSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the DoorSurface UML class; including the name, definition, type, and cardinality of those documented in the DoorSurface section of the Construction Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the DoorSurface UML class; including the name, definition, type, and cardinality of those documented in the DoorSurface section of the Construction Data Dictionary .
---	--

Class DoorSurface

Definition:

Subtype Of: [AbstractFillingSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [DoorSurface](#)

Target Role: address

Target Class: [Address](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [DoorSurface](#)

Target Role:

Target Class: [AbstractFillingSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Door](#)

Target Role: boundary

Target Class: [DoorSurface](#)

Attributes

12.27.13. Class DoorUsageValue

Subclass of <– section,>>

Requirement 260	/req/Construction/DoorUsageValue
A	Any use of the DoorUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorUsageValue UML class as documented in the DoorUsageValue section of the Construction Data Dictionary .

Class DoorUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.27.14. Class ElevationReferenceValue

Subclass of <-- section,>>

Requirement 261	/req/Construction/ElevationReferenceValue
A	Any use of the ElevationReferenceValue type in an Implementation Specification SHALL be restricted to the valid values specified in the ElevationReferenceValue UML class as documented in the ElevationReferenceValue section of the Construction Data Dictionary .

Class ElevationReferenceValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [ElevationReferenceValue](#)

Attributes

12.27.15. Class EventValue

Subclass of <-- section,>>

Requirement 262	/req/Construction/EventValue
A	Any use of the EventValue type in an Implementation Specification SHALL be restricted to the valid values specified in the EventValue UML class as documented in the EventValue section of the Construction Data Dictionary .

Class EventValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: Note

Target Role:

Target Class: EventValue

Attributes

12.27.16. Class FloorSurface

Subclass of AbstractConstructionSurface

Requirement 263	/req/Construction/FloorSurface
A	The Implementation Specification SHALL contain an element with the same definition as the FloorSurface UML class as documented in the FloorSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the FloorSurface UML class as documented in the FloorSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the FloorSurface UML class; including the name, definition, type, and cardinality of those documented in the FloorSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the FloorSurface UML class; including the name, definition, type, and cardinality of those documented in the FloorSurface section of the Construction Data Dictionary .

Class FloorSurface

Definition:

Subtype Of: AbstractConstructionSurface

Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	FloorSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Attributes	

12.27.17. Class GroundSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 264	/req/Construction/GroundSurface
A	The Implementation Specification SHALL contain an element with the same definition as the GroundSurface UML class as documented in the GroundSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GroundSurface UML class as documented in the GroundSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GroundSurface UML class; including the name, definition, type, and cardinality of those documented in the GroundSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GroundSurface UML class; including the name, definition, type, and cardinality of those documented in the GroundSurface section of the Construction Data Dictionary .

Class GroundSurface
Definition:
Subtype Of: AbstractConstructionSurface
Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GroundSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Attributes	

12.27.18. Class InteriorWallSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 265	/req/Construction/InteriorWallSurface
A	The Implementation Specification SHALL contain an element with the same definition as the InteriorWallSurface UML class as documented in the InteriorWallSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the InteriorWallSurface UML class as documented in the InteriorWallSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the InteriorWallSurface UML class; including the name, definition, type, and cardinality of those documented in the InteriorWallSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the InteriorWallSurface UML class; including the name, definition, type, and cardinality of those documented in the InteriorWallSurface section of the Construction Data Dictionary .

Class InteriorWallSurface
Definition:
Subtype Of: AbstractConstructionSurface
Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	InteriorWallSurface
Target Role:	
Target Class:	AbstractConstructionSurface
Attributes	

12.27.19. Class OtherConstruction

Subclass of [AbstractConstruction](#)

Requirement 266	/req/Construction/OtherConstruction
A	The Implementation Specification SHALL contain an element with the same definition as the OtherConstruction UML class as documented in the OtherConstruction section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the OtherConstruction UML class as documented in the OtherConstruction section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the OtherConstruction UML class; including the name, definition, type, and cardinality of those documented in the OtherConstruction section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the OtherConstruction UML class; including the name, definition, type, and cardinality of those documented in the OtherConstruction section of the Construction Data Dictionary .

Class OtherConstruction
Definition:
Subtype Of: AbstractConstruction
Stereotype: «TopLevelFeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	OtherConstruction
Target Role:	
Target Class:	AbstractConstruction
Attributes	
Attribute Name:	class
Value Type:	OtherConstructionClassValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	OtherConstructionFunctionValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	OtherConstructionUsageValue
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.27.20. Class OtherConstructionClassValue

Subclass of <-- section,>>

Requirement 267	/req/Construction/OtherConstructionClassValue
A	Any use of the OtherConstructionClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherConstructionClassValue UML class as documented in the OtherConstructionClassValue section of the Construction Data Dictionary .

Class OtherConstructionClassValue
Definition:
Subtype Of: <-- section,>>
Stereotype: «CodeList»
Associations
Attributes

12.27.21. Class OtherConstructionFunctionValue

Subclass of <-- section,>>

Requirement 268	/req/Construction/OtherConstructionFunctionValue
A	Any use of the OtherConstructionFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherConstructionFunctionValue UML class as documented in the OtherConstructionFunctionValue section of the Construction Data Dictionary .

Class OtherConstructionFunctionValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.27.22. Class OtherConstructionUsageValue

Subclass of <-- section,>>

Requirement 269	/req/Construction/OtherConstructionUsageValue
A	Any use of the OtherConstructionUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the OtherConstructionUsageValue UML class as documented in the OtherConstructionUsageValue section of the Construction Data Dictionary .

Class OtherConstructionUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.27.23. Class OuterCeilingSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 270	/req/Construction/OuterCeilingSurface
-----------------	---------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the OuterCeilingSurface UML class as documented in the OuterCeilingSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the OuterCeilingSurface UML class as documented in the OuterCeilingSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the OuterCeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterCeilingSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the OuterCeilingSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterCeilingSurface section of the Construction Data Dictionary .

Class OuterCeilingSurface

Definition:

Subtype Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [OuterCeilingSurface](#)

Target Role:

Target Class: [AbstractConstructionSurface](#)

Attributes

12.27.24. Class OuterFloorSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 271	/req/Construction/OuterFloorSurface
A	The Implementation Specification SHALL contain an element with the same definition as the OuterFloorSurface UML class as documented in the OuterFloorSurface section of the Construction Data Dictionary .

B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the OuterFloorSurface UML class as documented in the OuterFloorSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the OuterFloorSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterFloorSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the OuterFloorSurface UML class; including the name, definition, type, and cardinality of those documented in the OuterFloorSurface section of the Construction Data Dictionary .

Class OuterFloorSurface

Definition:

Subtype Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [OuterFloorSurface](#)

Target Role:

Target Class: [AbstractConstructionSurface](#)

Attributes

12.27.25. Class RoofSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 272	/req/Construction/RoofSurface
A	The Implementation Specification SHALL contain an element with the same definition as the RoofSurface UML class as documented in the RoofSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the RoofSurface UML class as documented in the RoofSurface section of the Construction Data Dictionary .

C	The implementation Specification SHALL represent the attributes of the RoofSurface UML class; including the name, definition, type, and cardinality of those documented in the RoofSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the RoofSurface UML class; including the name, definition, type, and cardinality of those documented in the RoofSurface section of the Construction Data Dictionary .

Class RoofSurface

Definition:

Subtype Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [RoofSurface](#)

Target Role:

Target Class: [AbstractConstructionSurface](#)

Attributes

12.27.26. Class WallSurface

Subclass of [AbstractConstructionSurface](#)

Requirement 273	/req/Construction/WallSurface
A	The Implementation Specification SHALL contain an element with the same definition as the WallSurface UML class as documented in the WallSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WallSurface UML class as documented in the WallSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the WallSurface UML class; including the name, definition, type, and cardinality of those documented in the WallSurface section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the WallSurface UML class; including the name, definition, type, and cardinality of those documented in the WallSurface section of the Construction Data Dictionary .

Class WallSurface

Definition:

Subtype Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [WallSurface](#)

Target Role:

Target Class: [AbstractConstructionSurface](#)

Attributes

12.27.27. Class Window

Subclass of [AbstractFillingElement](#)

Requirement 274	/req/Construction/Window
A	The Implementation Specification SHALL contain an element with the same definition as the Window UML class as documented in the Window section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Window UML class as documented in the Window section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Window UML class; including the name, definition, type, and cardinality of those documented in the Window section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Window UML class; including the name, definition, type, and cardinality of those documented in the Window section of the Construction Data Dictionary .

Class Window

Definition:

Subtype Of: [AbstractFillingElement](#)

Stereotype: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	Window
Target Role:	
Target Class:	AbstractFillingElement
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	Window
Target Role:	boundary
Target Class:	WindowSurface
Attributes	
Attribute Name:	class
Value Type:	WindowClassName
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	WindowFunctionName
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	WindowUsageName
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

12.27.28. Class [WindowClassName](#)

Subclass of <← section,>>

Requirement 275	/req/Construction/ WindowClassName
A	Any use of the WindowClassName type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowClassName UML class as documented in the Construction Data Dictionary .

Class [WindowClassName](#)

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.27.29. Class WindowFunctionValue

Subclass of <-- section,>>

Requirement 276	/req/Construction/WindowFunctionValue
A	Any use of the WindowFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowFunctionValue UML class as documented in the WindowFunctionValue section of the Construction Data Dictionary .

Class WindowFunctionValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Attributes

12.27.30. Class WindowSurface

Subclass of [AbstractFillingSurface](#)

Requirement 277	/req/Construction/WindowSurface
A	The Implementation Specification SHALL contain an element with the same definition as the WindowSurface UML class as documented in the WindowSurface section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the WindowSurface UML class as documented in the WindowSurface section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the WindowSurface UML class; including the name, definition, type, and cardinality of those documented in the WindowSurface section of the Construction Data Dictionary .

D	The implementation Specification SHALL represent the attributes of all parent classes of the the WindowSurface UML class; including the name, definition, type, and cardinality of those documented in the WindowSurface section of the Construction Data Dictionary .
---	--

Class WindowSurface

Definition:

Subtype Of: [AbstractFillingSurface](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [WindowSurface](#)

Target Role:

Target Class: [AbstractFillingSurface](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Window](#)

Target Role: boundary

Target Class: [WindowSurface](#)

Attributes

12.27.31. Class WindowUsageValue

Subclass of <-- section,>>

Requirement 278	/req/Construction/WindowUsageValue
A	Any use of the WindowUsageValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowUsageValue UML class as documented in the WindowUsageValue section of the Construction Data Dictionary .

Class WindowUsageValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Attributes

12.27.32. Class ConditionOfConstructionValue

Subclass of <-- section,>>

Requirement 279	/req/Construction/ConditionOfConstructionValue
A	Any use of the ConditionOfConstructionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the ConditionOfConstructionValue UML class as documented in the ConditionOfConstructionValue section of the Construction Data Dictionary .

Class ConditionOfConstructionValue

Definition:

Subtype Of: <-- section,>>

Stereotype:

Associations

Attributes

Attribute Name: declined

Value Type:

Definition: [INSPIRE] The construction cannot be used under normal conditions, though its main elements (walls, roof) are still present.

Multiplicity:

Stereotype:

Attribute Name: demolished

Value Type:

Definition: [INSPIRE] The construction has been demolished. There are no more visible remains.

Multiplicity:

Stereotype:

Attribute Name: functional

Value Type:

Definition: [INSPIRE] The construction is functional.

Multiplicity:

Stereotype:

Attribute Name: projected

Value Type:

Definition: [INSPIRE] The construction is being designed. Construction has not yet started.

Multiplicity:

Stereotype:

Attribute Name:	ruin
Value Type:	
Definition:	[INSPIRE] The construction has been partly demolished and some main elements (roof, walls) have been destroyed. There are some visible remains of the construction.
Multiplicity:	
Stereotype:	
Attribute Name:	underConstruction
Value Type:	
Definition:	[INSPIRE] The construction is under construction and not yet functional. This applies only to the initial construction of the construction and not to maintenance work.
Multiplicity:	
Stereotype:	

12.27.33. Class ConstructionEvent

Subclass of <-- section,>>

Requirement 280	/req/Construction/ConstructionEvent
A	Any use of the ConstructionEvent type in the Implementation Specification SHALL have the same definition as the ConstructionEvent UML class as documented in the ConstructionEvent section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the ConstructionEvent UML class as documented in the ConstructionEvent section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the ConstructionEvent UML class; including the name, definition, type, and cardinality of those documented in the ConstructionEvent section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the ConstructionEvent UML class; including the name, definition, type, and cardinality of those documented in the ConstructionEvent section of the Construction Data Dictionary .

Class ConstructionEvent

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Attributes

Attribute Name: dateOfEvent

Value Type: Date

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: description

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: event

Value Type: EventValue

Definition:

Multiplicity:

Stereotype: «Property»

12.27.34. Class Elevation

Subclass of <-->

Requirement 281	/req/Construction/Elevation
A	Any use of the Elevation type in the Implementation Specification SHALL have the same definition as the Elevation UML class as documented in the Elevation section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Elevation UML class as documented in the Elevation section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Elevation UML class; including the name, definition, type, and cardinality of those documented in the Elevation section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Elevation UML class; including the name, definition, type, and cardinality of those documented in the Elevation section of the Construction Data Dictionary .

Class Elevation

Definition:

Subtype Of: <-->

Stereotype: «DataType»

Associations

Attributes

Attribute Name:	elevationReference
Value Type:	ElevationReferenceValue
Definition:	[INSPIRE] Element where the elevation was measured.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	elevationValue
Value Type:	DirectPosition
Definition:	[INSPIRE] Value of the elevation.
Multiplicity:	
Stereotype:	«Property»

12.27.35. Class Height

Subclass of <-- section,>>

Requirement 282	/req/Construction/Height
A	Any use of the Height type in the Implementation Specification SHALL have the same definition as the Height UML class as documented in the Height section of the Construction Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Height UML class as documented in the Height section of the Construction Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Height UML class; including the name, definition, type, and cardinality of those documented in the Height section of the Construction Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the Height UML class; including the name, definition, type, and cardinality of those documented in the Height section of the Construction Data Dictionary .

Class Height

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Attributes

Attribute Name:	highReference
Value Type:	ElevationReferenceValue
Definition:	[INSPIRE] Element used as the high reference.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	lowReference
Value Type:	ElevationReferenceValue
Definition:	[INSPIRE] Element as the low reference.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	status
Value Type:	HeightStatusValue
Definition:	[INSPIRE] The way the height has been captured.
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	value
Value Type:	Length
Definition:	[INSPIRE] Value of the height above ground.
Multiplicity:	
Stereotype:	«Property»

12.27.36. Class HeightStatusValue

Subclass of <-- section,>>

Requirement 283	/req/Construction/HeightStatusValue
A	Any use of the HeightStatusValue type in an Implementation Specification SHALL be restricted to the valid values specified in the HeightStatusValue UML class as documented in the HeightStatusValue section of the Construction Data Dictionary .

Class HeightStatusValue
Definition:
Subtype Of: <-- section,>>
Stereotype:
Associations
Attributes
Attribute Name: estimated
Value Type:
Definition: [INSPIRE] The height has been estimated and not measured.
Multiplicity:
Stereotype:

Attribute Name: measured

Value Type:

Definition: [INSPIRE] The height has been (directly or indirectly) measured.

Multiplicity:

Stereotype:

12.27.37. Class RelationToConstruction

Subclass of <-- section,>>

Requirement 284	/req/Construction/RelationToConstruction
A	Any use of the RelationToConstruction type in an Implementation Specification SHALL be restricted to the valid values specified in the RelationToConstruction UML class as documented in the RelationToConstruction section of the Construction Data Dictionary .

Class RelationToConstruction

Definition:

Subtype Of: <-- section,>>

Stereotype:

Associations

Attributes

Attribute Name: inside

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: outside

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: bothInsideAndOutside

Value Type:

Definition:

Multiplicity:

Stereotype:

12.27.38. Additional Information

The following sections provide additional information which may not be readily available through the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

12.28. Dynamizer

Requirements Class

<http://www.opengis.net/spec/CityGML/3.1/req/req-class-dynamizer>

Target type	Conceptual Model
Dependency	TBD
Dependency	TBD

TBD

The UML diagram of the Dynamizer Model is depicted in [Dynamizer UML Diagram](#). The Data Dictionary for the Dynamizer Package is provided in section [Dynamizer Data Dictionary](#).

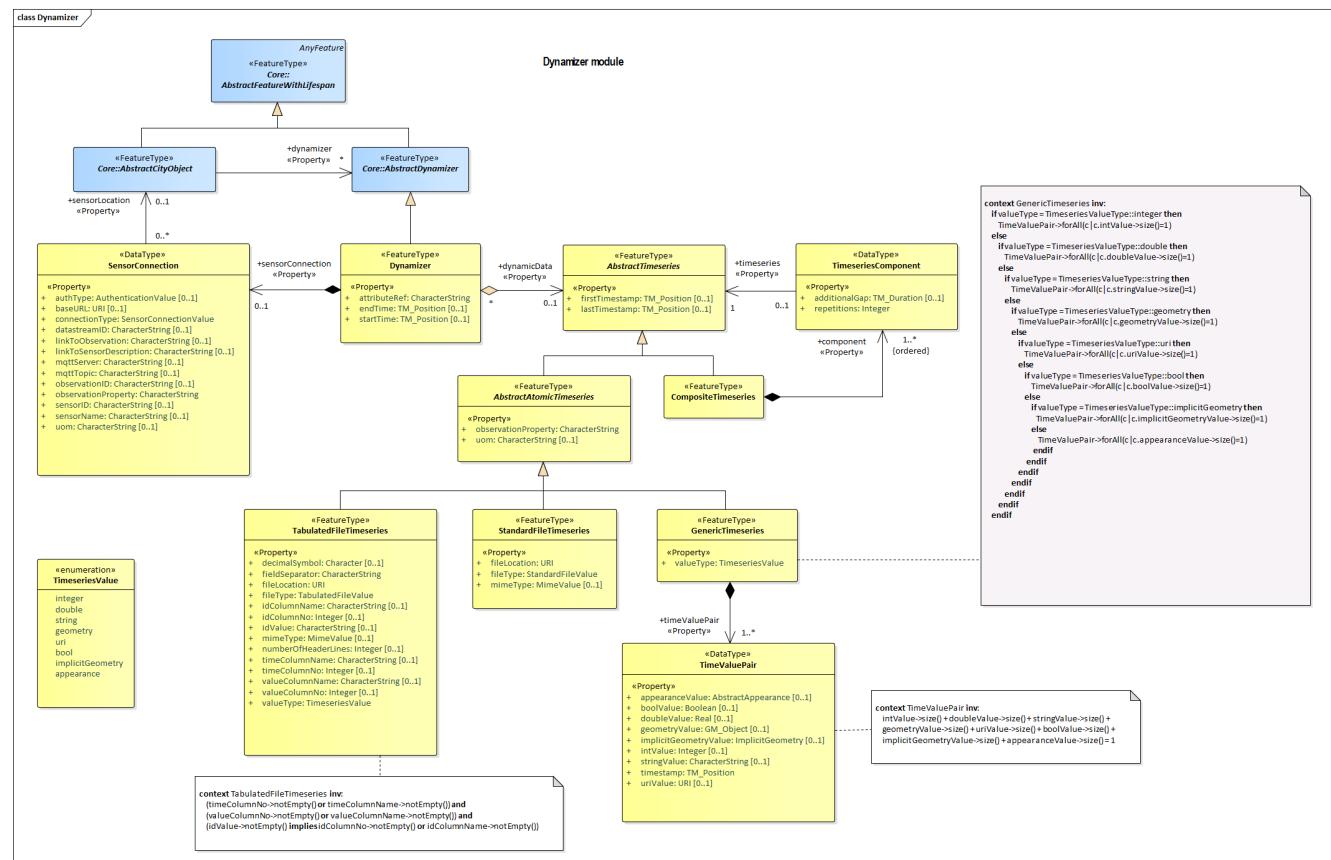


Figure 28. UML diagram of the Dynamizer Model.

The [UML diagram of the Dynamizer Model](#) is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

12.29. Dynamizer Model Data Dictionary

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

12.29.1. Class AbstractAtomicTimeseries

Subclass of [AbstractTimeseries](#)

Requirement 285	/req/Dynamizer/AbstractAtomicTimeSeries
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractAtomicTimeSeries UML class as documented in the AbstractAtomicTimeSeries section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractAtomicTimeSeries UML class as documented in the AbstractAtomicTimeSeries section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractAtomicTimeSeries UML class; including the name, definition, type, and cardinality of those documented in the AbstractAtomicTimeSeries section of the Dynamizer Data Dictionary .

Class AbstractAtomicTimeseries

Definition:

Subtype Of: [AbstractTimeseries](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractAtomicTimeseries](#)

Target Role:

Target Class: [AbstractTimeseries](#)

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	StandardFileTimeseries
Target Role:	
Target Class:	AbstractAtomicTimeseries
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	GenericTimeseries
Target Role:	
Target Class:	AbstractAtomicTimeseries
Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	TabulatedFileTimeseries
Target Role:	
Target Class:	AbstractAtomicTimeseries
Attributes	
Attribute Name:	observationProperty
Value Type:	CharacterString
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	uom
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.29.2. Class AbstractTimeseries

Subclass of <-- section,>>

Requirement 286	/req/Dynamizer/AbstractTimeSeries
A	The Implementation Specification SHALL contain an element which incorporates the concept defined in the AbstractTimeSeries UML class as documented in the AbstractTimeSeries section of the Dynamizer Data Dictionary .

B	The Implementation Specification SHALL provide an element which represents associations with the same source, target, direction, and cardinalities as those of the AbstractTimeSeries UML class as documented in the AbstractTimeSeries section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL provide an element which represents the attributes of the AbstractTimeSeries UML class; including the name, definition, type, and cardinality of those documented in the AbstractTimeSeries section of the Dynamizer Data Dictionary .

Class AbstractTimeseries

Definition:

Subtype Of: <-- section,>>

Stereotype: «FeatureType»

Associations

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [AbstractAtomicTimeseries](#)

Target Role:

Target Class: [AbstractTimeseries](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [TimeseriesComponent](#)

Target Role: timeseries

Target Class: [AbstractTimeseries](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Dynamizer](#)

Target Role: dynamicData

Target Class: [AbstractTimeseries](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [CompositeTimeseries](#)

Target Role:

Target Class: [AbstractTimeseries](#)

Attributes

Attribute Name: firstTimestamp

Value Type: TM_Position

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: lastTimestamp

Value Type: TM_Position

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

12.29.3. Class AuthenticationValue

Subclass of <-- section,>>

Requirement 287	/req/Dynamizer/AuthenticationValue
A	Any use of the AuthenticationValue type in an Implementation Specification SHALL be restricted to the valid values specified in the AuthenticationValue UML class as documented in the AuthenticationValue section of the Dynamizer Data Dictionary .

Class AuthenticationValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: Note

Target Role:

Target Class: AuthenticationValue

Attributes

12.29.4. Class CompositeTimeseries

Subclass of [AbstractTimeseries](#)

Requirement 288	/req/Dynamizer/CompositeTimeseries
-----------------	------------------------------------

A	The Implementation Specification SHALL contain an element with the same definition as the CompositeTimeseries UML class as documented in the CompositeTimeseries section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the CompositeTimeseries UML class as documented in the CompositeTimeseries section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the CompositeTimeseries UML class; including the name, definition, type, and cardinality of those documented in the CompositeTimeseries section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the CompositeTimeseries UML class; including the name, definition, type, and cardinality of those documented in the CompositeTimeseries section of the Dynamizer Data Dictionary .

Class CompositeTimeseries

Definition:

Subtype Of: [AbstractTimeseries](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [CompositeTimeseries](#)

Target Role: component

Target Class: [TimeseriesComponent](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [CompositeTimeseries](#)

Target Role:

Target Class: [AbstractTimeseries](#)

Attributes

12.29.5. Class Dynamizer

Subclass of [AbstractDynamizer](#)

Requirement 289	/req/Dynamizer/Dynamizer
A	The Implementation Specification SHALL contain an element with the same definition as the Dynamizer UML class as documented in the Dynamizer section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the Dynamizer UML class as documented in the Dynamizer section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the Dynamizer UML class; including the name, definition, type, and cardinality of those documented in the Dynamizer section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Dynamizer UML class; including the name, definition, type, and cardinality of those documented in the Dynamizer section of the Dynamizer Data Dictionary .

Class Dynamizer

Definition:

Subtype Of: [AbstractDynamizer](#)

Stereotype: «FeatureType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Dynamizer](#)

Target Role: dynamicData

Target Class: [AbstractTimeseries](#)

Name:

Type: Generalization

Direction: Source → Destination

Source Role:

Source Class: [Dynamizer](#)

Target Role:

Target Class: [AbstractDynamizer](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Dynamizer](#)

Target Role: sensorConnection

Target Class: [SensorConnection](#)

Attributes

Attribute Name:	attributeRef
Value Type:	CharacterString
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	endTime
Value Type:	TM_Position
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	startTime
Value Type:	TM_Position
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.29.6. Class GenericTimeseries

Subclass of [AbstractAtomicTimeseries](#)

Requirement 290	/req/Dynamizer/GenericTimeseries
A	The Implementation Specification SHALL contain an element with the same definition as the GenericTimeseries UML class as documented in the GenericTimeseries section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the GenericTimeseries UML class as documented in the GenericTimeseries section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the GenericTimeseries UML class; including the name, definition, type, and cardinality of those documented in the GenericTimeseries section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the GenericTimeseries UML class; including the name, definition, type, and cardinality of those documented in the GenericTimeseries section of the Dynamizer Data Dictionary .

Class GenericTimeseries

Definition:
 Subtype Of: [AbstractAtomicTimeseries](#)
 Stereotype: «FeatureType»

Associations

Name:
 Type: Generalization
 Direction: Source → Destination
 Source Role:
 Source Class: [GenericTimeseries](#)
 Target Role:
 Target Class: [AbstractAtomicTimeseries](#)

Name:
 Type: Association
 Direction: Source → Destination
 Source Role:
 Source Class: [GenericTimeseries](#)
 Target Role: timeValuePair
 Target Class: [TimeValuePair](#)

Name:
 Type: NoteLink
 Direction: Source → Destination
 Source Role:
 Source Class: [Note](#)
 Target Role:
 Target Class: [GenericTimeseries](#)

Attributes

Attribute Name: valueType
 Value Type: TimeseriesValue
 Definition:
 Multiplicity:
 Stereotype: «Property»

12.29.7. Class SensorConnectionValue

Subclass of <– section,>>

Requirement 291	/req/Dynamizer/SensorConnectionValue
A	Any use of the SensorConnectionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the SensorConnectionValue UML class as documented in the SensorConnectionValue section of the Dynamizer Data Dictionary .

Class SensorConnectionValue

Definition:
Subtype Of: <-- section,>>
StereoType: «CodeList»

Associations

Name:
Type: NoteLink
Direction: Source → Destination
Source Role:
Source Class: [Note](#)
Target Role:
Target Class: [SensorConnectionValue](#)

Attributes

12.29.8. Class StandardFileTimeseries

Subclass of [AbstractAtomicTimeseries](#)

Requirement 292	/req/Dynamizer/StandardFileTimeseries
A	The Implementation Specification SHALL contain an element with the same definition as the StandardFileTimeseries UML class as documented in the StandardFileTimeseries section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the StandardFileTimeseries UML class as documented in the StandardFileTimeseries section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the StandardFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the StandardFileTimeseries section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the StandardFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the StandardFileTimeseries section of the Dynamizer Data Dictionary .

Class StandardFileTimeseries

Definition:
Subtype Of: [AbstractAtomicTimeseries](#)
StereoType: «FeatureType»

Associations

Name:	
Type:	Generalization
Direction:	Source → Destination
Source Role:	
Source Class:	StandardFileTimeseries
Target Role:	
Target Class:	AbstractAtomicTimeseries

Attributes

Attribute Name:	fileLocation
Value Type:	URI
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	fileType
Value Type:	StandardFieldValue
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	MimeType
Value Type:	MimeType
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.29.9. Class StandardFieldValue

Subclass of <– section,>>

Requirement 293	/req/Dynamizer/StandardFieldValue
A	Any use of the StandardFieldValue type in an Implementation Specification SHALL be restricted to the valid values specified in the StandardFieldValue UML class as documented in the StandardFieldValue section of the Dynamizer Data Dictionary .

Class StandardFieldValue

Definition:
Subtype Of: <– section,>>
Stereotype: «CodeList»

Associations

Name:	
Type:	NoteLink
Direction:	Source → Destination
Source Role:	
Source Class:	Note
Target Role:	
Target Class:	StandardFileValue
Attributes	

12.29.10. Class TabulatedFileTimeseries

Subclass of [AbstractAtomicTimeseries](#)

Requirement 294	/req/Dynamizer/TabulatedFileTimeseries
A	The Implementation Specification SHALL contain an element with the same definition as the TabulatedFileTimeseries UML class as documented in the TabulatedFileTimeseries section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TabulatedFileTimeseries UML class as documented in the TabulatedFileTimeseries section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TabulatedFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the TabulatedFileTimeseries section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TabulatedFileTimeseries UML class; including the name, definition, type, and cardinality of those documented in the TabulatedFileTimeseries section of the Dynamizer Data Dictionary .

Class TabulatedFileTimeseries
Definition:
Subtype Of: AbstractAtomicTimeseries
Stereotype: «FeatureType»
Associations

<p>Name:</p> <p>Type: Generalization</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: TabulatedFileTimeseries</p> <p>Target Role:</p> <p>Target Class: AbstractAtomicTimeseries</p>
<p>Name:</p> <p>Type: NoteLink</p> <p>Direction: Source → Destination</p> <p>Source Role:</p> <p>Source Class: Note</p> <p>Target Role:</p> <p>Target Class: TabulatedFileTimeseries</p>
Attributes
<p>Attribute Name: decimalSymbol</p> <p>Value Type: Character</p> <p>Definition:</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: fieldSeparator</p> <p>Value Type: CharacterString</p> <p>Definition:</p> <p>Multiplicity:</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: fileLocation</p> <p>Value Type: URI</p> <p>Definition:</p> <p>Multiplicity:</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: fileType</p> <p>Value Type: TabulatedFileValue</p> <p>Definition:</p> <p>Multiplicity:</p> <p>Stereotype: «Property»</p>
<p>Attribute Name: idColumnName</p> <p>Value Type: CharacterString</p> <p>Definition:</p> <p>Multiplicity: [0..1]</p> <p>Stereotype: «Property»</p>

Attribute Name: idColumnNo

Value Type: Integer

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: idValue

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: mimeType

Value Type: MimeValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: numberofHeaderLines

Value Type: Integer

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: timeColumnName

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: timeColumnNo

Value Type: Integer

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: valueColumnName

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: valueColumnNo

Value Type: Integer

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	valueType
Value Type:	TimeseriesValue
Definition:	
Multiplicity:	
Stereotype:	«Property»

12.29.11. Class TabulatedFileValue

Subclass of <-- section,>>

Requirement 295	/req/Dynamizer/TabulatedFileValue
A	Any use of the TabulatedFileValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TabulatedFileValue UML class as documented in the TabulatedFileValue section of the Dynamizer Data Dictionary .

Class TabulatedFileValue

Definition:

Subtype Of: <-- section,>>

Stereotype: «CodeList»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [TabulatedFileValue](#)

Attributes

12.29.12. Class SensorConnection

Subclass of <-- section,>>

Requirement 296	/req/Dynamizer/SensorConnection
A	Any use of the SensorConnection type in the Implementation Specification SHALL have the same definition as the SensorConnection UML class as documented in the SensorConnection section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the SensorConnection UML class as documented in the SensorConnection section of the Dynamizer Data Dictionary .

C	The implementation Specification SHALL represent the attributes of the SensorConnection UML class; including the name, definition, type, and cardinality of those documented in the SensorConnection section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the SensorConnection UML class; including the name, definition, type, and cardinality of those documented in the SensorConnection section of the Dynamizer Data Dictionary .

Class SensorConnection

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [SensorConnection](#)

Target Role: sensorLocation

Target Class: [AbstractCityObject](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [Dynamizer](#)

Target Role: sensorConnection

Target Class: [SensorConnection](#)

Attributes

Attribute Name: authType

Value Type: AuthenticationValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: baseURL

Value Type: URI

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	connectionType
Value Type:	SensorConnectionValue
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	datastreamID
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	linkToObservation
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	linkToSensorDescription
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	mqttServer
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	mqttTopic
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	observationID
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	observationProperty
Value Type:	CharacterString
Definition:	
Multiplicity:	
Stereotype:	«Property»

Attribute Name:	sensorID
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	sensorName
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

Attribute Name:	uom
Value Type:	CharacterString
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

12.29.13. Class TimeseriesComponent

Subclass of <-- section,>>

Requirement 297	/req/Dynamizer/TimeseriesComponent
A	Any use of the TimeseriesComponent type in the Implementation Specification SHALL have the same definition as the TimeseriesComponent UML class as documented in the TimeseriesComponent section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TimeseriesComponent UML class as documented in the TimeseriesComponent section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TimeseriesComponent UML class; including the name, definition, type, and cardinality of those documented in the TimeseriesComponent section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TimeseriesComponent UML class; including the name, definition, type, and cardinality of those documented in the TimeseriesComponent section of the Dynamizer Data Dictionary .

Class TimeseriesComponent

Definition:	
Subtype Of:	<-- section,>>
Stereotype:	«DataType»

Associations	
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	TimeseriesComponent
Target Role:	timeseries
Target Class:	AbstractTimeseries
Name:	
Type:	Association
Direction:	Source → Destination
Source Role:	
Source Class:	CompositeTimeseries
Target Role:	component
Target Class:	TimeseriesComponent
Attributes	
Attribute Name:	additionalGap
Value Type:	TM_Duration
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	repetitions
Value Type:	Integer
Definition:	
Multiplicity:	
Stereotype:	«Property»

12.29.14. Class TimeseriesValue

Subclass of <-- section,>>

Requirement 298	/req/Dynamizer/TimeseriesValue
A	Any use of the TimeseriesValue type in an Implementation Specification SHALL be restricted to the valid values specified in the TimeseriesValue UML class as documented in the Dynamizer Data Dictionary .

Class TimeseriesValue	
Definition:	
Subtype Of:	<-- section,>>
Stereotype:	
Associations	
Attributes	

Attribute Name: integer

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: double

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: string

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: geometry

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: uri

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: bool

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: implicitGeometry

Value Type:

Definition:

Multiplicity:

Stereotype:

Attribute Name: appearance

Value Type:

Definition:

Multiplicity:

Stereotype:

12.29.15. Class TimeValuePair

Subclass of <– section,>>

Requirement 299	/req/Dynamizer/TimeValuePair
A	Any use of the TimeValuePair type in the Implementation Specification SHALL have the same definition as the TimeValuePair UML class as documented in the TimeValuePair section of the Dynamizer Data Dictionary .
B	The Implementation Specification SHALL represent associations with the same source, target, direction, and cardinalities as those of the TimeValuePair UML class as documented in the TimeValuePair section of the Dynamizer Data Dictionary .
C	The implementation Specification SHALL represent the attributes of the TimeValuePair UML class; including the name, definition, type, and cardinality of those documented in the TimeValuePair section of the Dynamizer Data Dictionary .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the TimeValuePair UML class; including the name, definition, type, and cardinality of those documented in the TimeValuePair section of the Dynamizer Data Dictionary .

Class TimeValuePair

Definition:

Subtype Of: <-- section,>>

Stereotype: «DataType»

Associations

Name:

Type: NoteLink

Direction: Source → Destination

Source Role:

Source Class: [Note](#)

Target Role:

Target Class: [TimeValuePair](#)

Name:

Type: Association

Direction: Source → Destination

Source Role:

Source Class: [GenericTimeseries](#)

Target Role: timeValuePair

Target Class: [TimeValuePair](#)

Attributes

Attribute Name: appearanceValue

Value Type: AbstractAppearance

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: boolValue

Value Type: Boolean

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: doubleValue

Value Type: Real

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: geometryValue

Value Type: GM_Object

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: implicitGeometryValue

Value Type: ImplicitGeometry

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: intValue

Value Type: Integer

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: stringValue

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: timestamp

Value Type: TM_Position

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: uriValue

Value Type: URI

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

12.29.16. Additional Information

The following sections provide additional information which may not be readily available through

the UML Model.

A detailed discussion of this Requirements Class can be found in the CityGML Best Practices document [here](#).

Chapter 13. Media Types for any data encoding(s)

A section describing the MIME-types to be used is mandatory for any standard involving data encodings. If no suitable MIME type exists in <http://www.iana.org/assignments/media-types/index.html> then this section may be used to define a new MIME type for registration with IANA.

Annex A: Conformance Class Abstract Test Suite (Normative)

A.1. Conformance Class Appearance

Abstract Test 1	/ats/Appearance/AbstractSurfaceData
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-AbstractSurfaceData
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractSurfaceData abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractSurfaceData abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractSurfaceData abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 2	/ats/Appearance/AbstractTexture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-AbstractTexture
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the AbstractTexture abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTexture abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTexture abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 3	/ats/Appearance/AbstractTextureParameterization
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-AbstractTextureParameterization
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTextureParameterization abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTextureParameterization abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTextureParameterization abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 4	/ats/Appearance/Appearance
------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-Appearance
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Appearance class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Appearance class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Appearance class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Appearance class as documented in the Conceptual Model.

Abstract Test 5	/ats/Appearance/Color
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-Color
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Color type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Color type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the Color type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Color type as documented in the Conceptual Model.

Abstract Test 6	/ats/Appearance/ColorPlusOpacity
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-ColorPlusOpacity
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ColorPlusOpacity type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ColorPlusOpacity type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ColorPlusOpacity type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ColorPlusOpacity type as documented in the Conceptual Model.

Abstract Test 7	/ats/Appearance/GeoreferencedTexture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Appearance/rc-GeoreferencedTexture
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GeoreferencedTexture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GeoreferencedTexture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GeoreferencedTexture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GeoreferencedTexture class as documented in the Conceptual Model.

Abstract Test 8	/ats/Appearance/ParameterizedTexture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-ParameterizedTexture
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ParameterizedTexture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ParameterizedTexture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the ParameterizedTexture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ParameterizedTexture class as documented in the Conceptual Model.

Abstract Test 9	/ats/Appearance/TexCoordGen
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-TexCoordGen
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TexCoordGen type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TexCoordGen type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TexCoordGen type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TexCoordGen type as documented in the Conceptual Model.

Abstract Test 10	/ats/Appearance/TexCoordList
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-TexCoordList
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TexCoordList type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TexCoordList type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TexCoordList type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TexCoordList type as documented in the Conceptual Model.

Abstract Test 11	/ats/Appearance/TextureAssociation
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-TextureAssociation
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TextureAssociation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TextureAssociation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TextureAssociation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TextureAssociation class as documented in the Conceptual Model.

Abstract Test 12	/ats/Appearance/TextureType
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-TextureType
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TextureType type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TextureType type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TextureType type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TextureType type as documented in the Conceptual Model.

Abstract Test 13	/ats/Appearance/WrapMode
-------------------------	---------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-WrapMode
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WrapMode type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WrapMode type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WrapMode type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the WrapMode type as documented in the Conceptual Model.

Abstract Test 14	/ats/Appearance/X3DMaterial
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Appearance/rc-X3DMaterial
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the X3DMaterial class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the X3DMaterial class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the X3DMaterial class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the X3DMaterial class as documented in the Conceptual Model.

A.2. Conformance Class Bridge

Abstract Test 15	/ats/Bridge/AbstractBridge
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-AbstractBridge
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractBridge abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractBridge abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractBridge abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 16	/ats/Bridge/Bridge
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-Bridge

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Bridge class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Bridge class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Bridge class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Bridge class as documented in the Conceptual Model.

Abstract Test 17	/ats/Bridge/BridgeClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 18	/ats/Bridge/BridgeConstructiveElement
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeConstructiveElement
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the BridgeConstructiveElement class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgeConstructiveElement class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeConstructiveElement class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgeConstructiveElement class as documented in the Conceptual Model.

Abstract Test 19	/ats/Bridge/BridgeConstructiveElementClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeConstructiveElementClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeConstructiveElementClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 20	/ats/Bridge/BridgeConstructiveElementFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeConstructiveElementFunctionValue
Test Method	Manual Inspection

A	Validate that all instances of the BridgeConstructiveElementFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 21	/ats/Bridge/BridgeConstructiveElementUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeConstructiveElementUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeConstructiveElementUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 22	/ats/Bridge/BridgeFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 23	/ats/Bridge/BridgeFurniture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeFurniture
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the BridgeFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgeFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgeFurniture class as documented in the Conceptual Model.

Abstract Test 24	/ats/Bridge/BridgeFurnitureClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeFurnitureClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 25	/ats/Bridge/BridgeFurnitureFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeFurnitureFunctionValue
Test Method	Manual Inspection

A	Validate that all instances of the BridgeFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 26	/ats/Bridge/BridgeFurnitureUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeFurnitureUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 27	/ats/Bridge/BridgeInstallation
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeInstallation
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BridgeInstallation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgeInstallation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeInstallation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the BridgeInstallation class as documented in the Conceptual Model.
---	---

Abstract Test 28	/ats/Bridge/BridgeInstallationClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeInstallationClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeInstallationClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 29	/ats/Bridge/BridgeInstallationFundntionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeInstallationFundntionValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeInstallationFundntionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 30	/ats/Bridge/BridgeInstallationUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeInstallationUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeInstallationUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 31	/ats/Bridge/BridgePart
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgePart
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BridgePart class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BridgePart class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgePart class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgePart class as documented in the Conceptual Model.

Abstract Test 32	/ats/Bridge/BridgeRoom
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeRoom
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BridgeRoom class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the BridgeRoom class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BridgeRoom class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BridgeRoom class as documented in the Conceptual Model.

Abstract Test 33	/ats/Bridge/BridgeRoomClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeRoomClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeRoomClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 34	/ats/Bridge/BridgeRoomFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeRoomFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeRoomFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 35	/ats/Bridge/BridgeRoomUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeRoomUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeRoomUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 36	/ats/Bridge/BridgeUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Bridge/rc-BridgeUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BridgeUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

A.3. Conformance Class Building

Abstract Test 37	/ats/Building/AbstractBuilding
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Building/rc-AbstractBuilding
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractBuilding abstract class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the AbstractBuilding abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractBuilding abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 38	/ats/Building/AbstractBuildingSubdivision
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Building/rc-AbstractBuildingSubdivision
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractBuildingSubdivision abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractBuildingSubdivision abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractBuildingSubdivision abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 39	/ats/Building/Building
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Building/rc-Building
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Building class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Building class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Building class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Building class as documented in the Conceptual Model.

Abstract Test 40	/ats/Building/BuildingClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 41	/ats/Building/BuildingConstructiveElement
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingConstructiveElement

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingConstructiveElement class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingConstructiveElement class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingConstructiveElement class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingConstructiveElement class as documented in the Conceptual Model.

Abstract Test 42	/ats/Building/BuildingConstructiveElementClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingConstructiveElementClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingConstructiveElementClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 43	/ats/Building/BuildingConstructiveElementFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingConstructiveElementFunctionValue

Test Method	Manual Inspection
A	Validate that all instances of the BuildingConstructiveElementFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 44	/ats/Building/BuildingConstructiveElementUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingConstructiveElementUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingConstructiveElementUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 45	/ats/Building/BuildingFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 46	/ats/Building/BuildingFurniture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingFurniture
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the BuildingFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingFurniture class as documented in the Conceptual Model.

Abstract Test 47	/ats/Building/BuildingFurnitureClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<i>/req/Building/rc-BuildingFurnitureClassValue</i>
Test Method	Manual Inspection
A	Validate that all instances of the BuildingFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 48	/ats/Building/BuildingFurnitureFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<i>/req/Building/rc-BuildingFurnitureFunctionValue</i>
Test Method	Manual Inspection

A	Validate that all instances of the BuildingFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 49	/ats/Building/BuildingFurnitureUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingFurnitureUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 50	/ats/Building/BuildingInstallation
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingInstallation
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingInstallation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingInstallation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingInstallation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the BuildingInstallation class as documented in the Conceptual Model.
---	---

Abstract Test 51	/ats/Building/BuildingInstallationClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingInstallationClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingInstallationClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 52	/ats/Building/BuildingInstallationFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingInstallationFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingInstallationFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 53	/ats/Building/BuildingInstallationUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingInstallationUsageValue
Test Method	Manual Inspection

A	Validate that all instances of the BuildingInstallationUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 54	/ats/Building/BuildingPart
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingPart
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingPart class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingPart class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingPart class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingPart class as documented in the Conceptual Model.

Abstract Test 55	/ats/Building/BuildingRoom
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingRoom
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingRoom class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the BuildingRoom class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingRoom class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingRoom class as documented in the Conceptual Model.

Abstract Test 56	/ats/Building/BuildingRoomClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingRoomClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingRoomClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 57	/ats/Building/BuildingRoomFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingRoomFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingRoomFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 58	/ats/Building/BuildingRoomUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingRoomUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingRoomUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 59	/ats/Building/BuildingSubdivisionClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingSubdivisionClassValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingSubdivisionClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 60	/ats/Building/BuildingSubdivisionFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingSubdivisionFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingSubdivisionFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 61	/ats/Building/BuildingSubdivisionUsageValue
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingSubdivisionUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingSubdivisionUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 62	/ats/Building/BuildingUnit
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-BuildingUnit
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BuildingUnit class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the BuildingUnit class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BuildingUnit class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BuildingUnit class as documented in the Conceptual Model.

Abstract Test 63	/ats/Building/BuildingUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	/req/Building/rc-BuildingUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the BuildingUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 64	/ats/Building/RoofTypeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-RoofTypeValue
Test Method	Manual Inspection
A	Validate that all instances of the RoofTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 65	/ats/Building/RoomElevationReferenceValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Building/rc-RoomElevationReferenceValue
Test Method	Manual Inspection
A	Validate that all instances of the RoomElevationReferenceValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 66	/ats/Building/RoomHeight
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Building/rc-RoomHeight

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the RoomHeight type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the RoomHeight type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the RoomHeight type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the RoomHeight type as documented in the Conceptual Model.

Abstract Test 67	/ats/Building/Storey
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Building/rc-Storey
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Storey class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Storey class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Storey class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the Storey class as documented in the Conceptual Model.
---	---

A.4. Conformance Class CityFurniture

Abstract Test 68	/ats/CityFurniture/CityFurniture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/CityFurniture/rc-CityFurniture
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the CityFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityFurniture class as documented in the Conceptual Model.

Abstract Test 69	/ats/CityFurniture/CityFurnitureClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/CityFurniture/rc-CityFurnitureClassValue
Test Method	Manual Inspection

A	Validate that all instances of the CityFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--

Abstract Test 70	/ats/CityFurniture/CityFurnitureFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/CityFurniture/rc-CityFurnitureFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the CityFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 71	/ats/CityFurniture/CityFurnitureUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/CityFurniture/rc-CityFurnitureUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the CityFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

A.5. Conformance Class CityObjectGroup

Abstract Test 72	/ats/CityObjectGroup/CityObjectGroup
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/CityObjectGroup/rc-CityObjectGroup
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the CityObjectGroup class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityObjectGroup class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityObjectGroup class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityObjectGroup class as documented in the Conceptual Model.

Abstract Test 73	/ats/CityObjectGroup/CityObjectGroupClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/CityObjectGroup/rc-CityObjectGroupClassValue
Test Method	Manual Inspection
A	Validate that all instances of the CityObjectGroupClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 74	/ats/CityObjectGroup/CityObjectGroupFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/CityObjectGroup/rc-CityObjectGroupFunctionValue
Test Method	Manual Inspection

A	Validate that all instances of the CityObjectGroupFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 75	/ats/CityObjectGroup/CityObjectGroupUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/CityObjectGroup/rc-CityObjectGroupUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the CityObjectGroupUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 76	/ats/CityObjectGroup/Role
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/CityObjectGroup/rc-Role
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Role class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Role class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Role class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the Role class as documented in the Conceptual Model.
---	---

A.6. Conformance Class Core

Abstract Test 77	/ats/Core/AbstractAppearance
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractAppearance
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractAppearance abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractAppearance abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractAppearance abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 78	/ats/Core/AbstractCityObject
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractCityObject
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the AbstractCityObject abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractCityObject abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractCityObject abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 79	/ats/Core/AbstractDynamizer
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractDynamizer
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractDynamizer abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractDynamizer abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractDynamizer abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 80	/ats/Core/AbstractFeatureWithLifespan
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractFeatureWithLifespan
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractFeatureWithLifespan abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractFeatureWithLifespan abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractFeatureWithLifespan abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 81	/ats/Core/AbstractGenericAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractGenericAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractGenericAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractGenericAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the AbstractGenericAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the AbstractGenericAttribute type as documented in the Conceptual Model.

Abstract Test 82	/ats/Core/AbstractLogicalSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractLogicalSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractLogicalSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractLogicalSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractLogicalSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 83	/ats/Core/AbstractOccupiedSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractOccupiedSpace

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractOccupiedSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractOccupiedSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractOccupiedSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 84	/ats/Core/AbstractPhysicalSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractPhysicalSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractPhysicalSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractPhysicalSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractPhysicalSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 85	/ats/Core/AbstractPointCloud
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractPointCloud
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractPointCloud abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractPointCloud abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractPointCloud abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 86	/ats/Core/AbstractSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the AbstractSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
---	--

Abstract Test 87	/ats/Core/AbstractSpaceBoundary
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractSpaceBoundary
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractSpaceBoundary abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractSpaceBoundary abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractSpaceBoundary abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 88	/ats/Core/AbstractThematicSurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractThematicSurface
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the AbstractThematicSurface abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractThematicSurface abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractThematicSurface abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 89	/ats/Core/AbstractUnoccupiedSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractUnoccupiedSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractUnoccupiedSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractUnoccupiedSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractUnoccupiedSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 90	/ats/Core/AbstractVersion
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractVersion
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractVersion abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractVersion abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractVersion abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 91	/ats/Core/AbstractVersionTransition
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Core/rc-AbstractVersionTransition
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractVersionTransition abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractVersionTransition abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the AbstractVersionTransition abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
---	--

Abstract Test 92	/ats/Core/Address
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Core/rc-Address
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Address class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Address class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Address class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Address class as documented in the Conceptual Model.

Abstract Test 93	/ats/Core/CityModel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Core/rc-CityModel
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the CityModel class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityModel class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityModel class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CityModel class as documented in the Conceptual Model.

Abstract Test 94	/ats/Core/CityObjectRelation
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-CityObjectRelation
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the CityObjectRelation type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CityObjectRelation type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CityObjectRelation type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the CityObjectRelation type as documented in the Conceptual Model.
---	--

Abstract Test 95	/ats/Core/ClosureSurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Core/rc-ClosureSurface
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ClosureSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ClosureSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ClosureSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ClosureSurface class as documented in the Conceptual Model.

Abstract Test 96	/ats/Core/DoubleBetween0and1
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-DoubleBetween0and1
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the DoubleBetween0and1 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleBetween0and1 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleBetween0and1 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DoubleBetween0and1 type as documented in the Conceptual Model.

Abstract Test 97	/ats/Core/DoubleBetween0and1List
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-DoubleBetween0and1List
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the DoubleBetween0and1List type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleBetween0and1List type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleBetween0and1List type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the DoubleBetween0and1List type as documented in the Conceptual Model.
---	--

Abstract Test 98	/ats/Core/DoubleList
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-DoubleList
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the DoubleList type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleList type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleList type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DoubleList type as documented in the Conceptual Model.

Abstract Test 99	/ats/Core/ExternalReference
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-ExternalReference
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the ExternalReference type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ExternalReference type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ExternalReference type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ExternalReference type as documented in the Conceptual Model.

Abstract Test 100	/ats/Core/ImplicitGeometry
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Core/rc-ImplicitGeometry
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ImplicitGeometry class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ImplicitGeometry class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ImplicitGeometry class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the ImplicitGeometry class as documented in the Conceptual Model.
---	---

Abstract Test 101	/ats/Core/IntegerBetween0and3
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-IntegerBetween0and3
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the IntegerBetween0and3 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the IntegerBetween0and3 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the IntegerBetween0and3 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the IntegerBetween0and3 type as documented in the Conceptual Model.

Abstract Test 102	/ats/Core/IntervalValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-IntervalValue
Test Method	Manual Inspection

A	Validate that all instances of the IntervalValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--

Abstract Test 103	/ats/Core/MimeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-MimeValue
Test Method	Manual Inspection
A	Validate that all instances of the MimeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 104	/ats/Core/Occupancy
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-Occupancy
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Occupancy type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Occupancy type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Occupancy type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the Occupancy type as documented in the Conceptual Model.
---	---

Abstract Test 105	/ats/Core/OccupantTypeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<i>/req/Core/rc-OccupantTypeValue</i>
Test Method	Manual Inspection
A	Validate that all instances of the OccupantTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 106	/ats/Core/OtherRelationTypeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<i>/req/Core/rc-OtherRelationTypeValue</i>
Test Method	Manual Inspection
A	Validate that all instances of the OtherRelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 107	/ats/Core/QualifiedArea
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<i>/req/Core/rc-QualifiedArea</i>
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the QualifiedArea type in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the QualifiedArea type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the QualifiedArea type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the QualifiedArea type as documented in the Conceptual Model.

Abstract Test 108	/ats/Core/QualifiedAreaValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-QualifiedAreaValue
Test Method	Manual Inspection
A	Validate that all instances of the QualifiedAreaValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 109	/ats/Core/QualifiedVolume
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-QualifiedVolume
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the QualifiedVolume type in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the QualifiedVolume type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the QualifiedVolume type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the QualifiedVolume type as documented in the Conceptual Model.

Abstract Test 110	/ats/Core/QualifiedVolumeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-QualifiedVolumeValue
Test Method	Manual Inspection
A	Validate that all instances of the QualifiedVolumeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 111	/ats/Core/RelationTypeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-RelationTypeValue
Test Method	Manual Inspection
A	Validate that all instances of the RelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 112	/ats/Core/RelativeToTerrain
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-RelativeToTerrain
Test Method	Manual Inspection
A	Validate that all instances of the RelativeToTerrain codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 113	/ats/Core/RelativeToWater
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-RelativeToWater
Test Method	Manual Inspection
A	Validate that all instances of the RelativeToWater codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 114	/ats/Core/SpaceType
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-SpaceType
Test Method	Manual Inspection
A	Validate that all instances of the SpaceType codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 115	/ats/Core/TemporalRelationTypeValue
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-TemporalRelationTypeValue
Test Method	Manual Inspection
A	Validate that all instances of the TemporalRelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 116	/ats/Core/TopologicalRelationTypeValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Core/rc-TopologicalRelationTypeValue
Test Method	Manual Inspection
A	Validate that all instances of the TopologicalRelationTypeValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 117	/ats/Core/TransformationMatrix2x2
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-TransformationMatrix2x2
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TransformationMatrix2x2 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TransformationMatrix2x2 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TransformationMatrix2x2 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TransformationMatrix2x2 type as documented in the Conceptual Model.

Abstract Test 118	/ats/Core/TransformationMatrix3x4
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-TransformationMatrix3x4
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TransformationMatrix3x4 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TransformationMatrix3x4 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TransformationMatrix3x4 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TransformationMatrix3x4 type as documented in the Conceptual Model.

Abstract Test 119	/ats/Core/TransformationMatrix4x4
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-TransformationMatrix4x4
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TransformationMatrix4x4 type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TransformationMatrix4x4 type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TransformationMatrix4x4 type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TransformationMatrix4x4 type as documented in the Conceptual Model.

Abstract Test 120	/ats/Core/XALAddressDetails
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Core/rc-XALAddressDetails
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the XALAddressDetails type in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the XALAddressDetails type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the XALAddressDetails type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the XALAddressDetails type as documented in the Conceptual Model.

A.7. Conformance Class Dynamizer

Abstract Test 121	/ats/Dynamizer/AbstractAtomicTimeSeries
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-AbstractAtomicTimeSeries
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractAtomicTimeSeries abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractAtomicTimeSeries abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractAtomicTimeSeries abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 122	/ats/Dynamizer/AbstractTimeSeries
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-AbstractTimeSeries
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTimeSeries abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTimeSeries abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTimeSeries abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 123	/ats/Dynamizer/AuthenticationValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-AuthenticationValue
Test Method	Manual Inspection
A	Validate that all instances of the AuthenticationValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 124	/ats/Dynamizer/CompositeTimeseries
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Dynamizer/rc-CompositeTimeseries
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the CompositeTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the CompositeTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the CompositeTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the CompositeTimeseries class as documented in the Conceptual Model.

Abstract Test 125	/ats/Dynamizer/Dynamizer
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-Dynamizer
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Dynamizer class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Dynamizer class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the Dynamizer class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Dynamizer class as documented in the Conceptual Model.

Abstract Test 126	/ats/Dynamizer/GenericTimeseries
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-GenericTimeseries
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericTimeseries class as documented in the Conceptual Model.

Abstract Test 127	/ats/Dynamizer/SensorConnection
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model

Requirement	/req/Dynamizer/rc-SensorConnection
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the SensorConnection type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the SensorConnection type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the SensorConnection type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the SensorConnection type as documented in the Conceptual Model.

Abstract Test 128	/ats/Dynamizer/SensorConnectionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-SensorConnectionValue
Test Method	Manual Inspection
A	Validate that all instances of the SensorConnectionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 129	/ats/Dynamizer/StandardFileTimeseries
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-StandardFileTimeseries

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the StandardFileTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the StandardFileTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the StandardFileTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the StandardFileTimeseries class as documented in the Conceptual Model.

Abstract Test 130	/ats/Dynamizer/StandardFieldValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-StandardFieldValue
Test Method	Manual Inspection
A	Validate that all instances of the StandardFieldValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 131	/ats/Dynamizer/TabulatedFileTimeseries
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-TabulatedFileTimeseries
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the TabulatedFileTimeseries class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TabulatedFileTimeseries class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TabulatedFileTimeseries class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TabulatedFileTimeseries class as documented in the Conceptual Model.

Abstract Test 132	/ats/Dynamizer/TabulatedFileValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-TabulatedFileValue
Test Method	Manual Inspection
A	Validate that all instances of the TabulatedFileValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 133	/ats/Dynamizer/TimeseriesComponent
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-TimeseriesComponent
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the TimeseriesComponent type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TimeseriesComponent type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TimeseriesComponent type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TimeseriesComponent type as documented in the Conceptual Model.

Abstract Test 134	/ats/Dynamizer/TimeseriesValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-TimeseriesValue
Test Method	Manual Inspection
A	Validate that all instances of the TimeseriesValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 135	/ats/Dynamizer/TimeValuePair
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Dynamizer/rc-TimeValuePair
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the TimeValuePair type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TimeValuePair type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TimeValuePair type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TimeValuePair type as documented in the Conceptual Model.

A.8. Conformance Class Generics

Abstract Test 136	/ats/Generics/DateAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-DateAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the DateAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DateAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the DateAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DateAttribute type as documented in the Conceptual Model.

Abstract Test 137	/ats/Generics/DoubleAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-DoubleAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the DoubleAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the DoubleAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the DoubleAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the DoubleAttribute type as documented in the Conceptual Model.

Abstract Test 138	/ats/Generics/GenericAttributeSet
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericAttributeSet
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericAttributeSet type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericAttributeSet type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericAttributeSet type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericAttributeSet type as documented in the Conceptual Model.

Abstract Test 139	/ats/Generics/GenericLogicalSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericLogicalSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericLogicalSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericLogicalSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the GenericLogicalSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericLogicalSpace class as documented in the Conceptual Model.

Abstract Test 140	/ats/Generics/GenericLogicalSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericLogicalSpaceClassValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericLogicalSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 141	/ats/Generics/GenericLogicalSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericLogicalSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericLogicalSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 142	/ats/Generics/GenericLogicalSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	/req/Generics/rc-GenericLogicalSpaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericLogicalSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 143	/ats/Generics/GenericOccupiedSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericOccupiedSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericOccupiedSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericOccupiedSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericOccupiedSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericOccupiedSpace class as documented in the Conceptual Model.

Abstract Test 144	/ats/Generics/GenericOccupiedSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericOccupiedSpaceClassValue

Test Method	Manual Inspection
A	Validate that all instances of the GenericOccupiedSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 145	/ats/Generics/GenericOccupiedSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericOccupiedSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericOccupiedSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 146	/ats/Generics/GenericOccupiedSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericOccupiedSpaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericOccupiedSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 147	/ats/Generics/GenericThematicSurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericThematicSurface
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the GenericThematicSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericThematicSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericThematicSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the GenericThematicSurface class as documented in the Conceptual Model.

Abstract Test 148	/ats/Generics/GenericThematicSurfaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericThematicSurfaceClassValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericThematicSurfaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 149	/ats/Generics/GenericThematicSurfaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericThematicSurfaceFunctionValue
Test Method	Manual Inspection

A	Validate that all instances of the GenericThematicSurfaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--

Abstract Test 150	/ats/Generics/GenericThematicSurfaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericThematicSurfaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericThematicSurfaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 151	/ats/Generics/GenericUnoccupiedSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericUnoccupiedSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the GenericUnoccupiedSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the GenericUnoccupiedSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the GenericUnoccupiedSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the GenericUnoccupiedSpace class as documented in the Conceptual Model.
---	---

Abstract Test 152	/ats/Generics/GenericUnoccupiedSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericUnoccupiedSpaceClassValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericUnoccupiedSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 153	/ats/Generics/GenericUnoccupiedSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericUnoccupiedSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the GenericUnoccupiedSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 154	/ats/Generics/GenericUnoccupiedSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Generics/rc-GenericUnoccupiedSpaceUsageValue
Test Method	Manual Inspection

A	Validate that all instances of the GenericUnoccupiedSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 155	/ats/Generics/IntAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-IntAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the IntAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the IntAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the IntAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the IntAttribute type as documented in the Conceptual Model.

Abstract Test 156	/ats/Generics/MeasureAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-MeasureAttribute
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the MeasureAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the MeasureAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the MeasureAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the MeasureAttribute type as documented in the Conceptual Model.

Abstract Test 157	/ats/Generics/StringAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-StringAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the StringAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the StringAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the StringAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the StringAttribute type as documented in the Conceptual Model.
---	---

Abstract Test 158	/ats/Generics/UriAttribute
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Generics/rc-UriAttribute
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the UriAttribute type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the UriAttribute type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the UriAttribute type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the UriAttribute type as documented in the Conceptual Model.

A.9. Conformance Class LandUse

Abstract Test 159	/ats/LandUse/LandUse
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/LandUse/rc-LandUse
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the LandUse class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the LandUse class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the LandUse class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the LandUse class as documented in the Conceptual Model.

Abstract Test 160	/ats/LandUse/LandUseClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/LandUse/rc-LandUseClassValue
Test Method	Manual Inspection
A	Validate that all instances of the LandUseClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 161	/ats/LandUse/LandUseFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/LandUse/rc-LandUseFunctionValue
Test Method	Manual Inspection

A	Validate that all instances of the LandUseFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 162	/ats/LandUse/LandUseUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/LandUse/rc-LandUseUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the LandUseUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

A.10. Conformance Class PointCloud

Abstract Test 163	/ats/PointCloud/PointCloud
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/PointCloud/rc-PointCloud
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the PointCloud class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the PointCloud class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the PointCloud class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

D	Validate that the properties of the data element include those of all parent classes of the PointCloud class as documented in the Conceptual Model.
---	---

A.11. Conformance Class Relief

Abstract Test 164	/ats/Relief/AbstractReliefComponent
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Relief/rc-AbstractReliefComponent
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractReliefComponent abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractReliefComponent abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractReliefComponent abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 165	/ats/Relief/BreaklineRelief
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Relief/rc-BreaklineRelief
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the BreaklineRelief class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the BreaklineRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the BreaklineRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the BreaklineRelief class as documented in the Conceptual Model.

Abstract Test 166	/ats/Relief/MassPointRelief
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Relief/rc-MassPointRelief
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the MassPointRelief class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the MassPointRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the MassPointRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the MassPointRelief class as documented in the Conceptual Model.

Abstract Test 167	/ats/Relief/RasterRelief
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Relief/rc-RasterRelief
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the RasterRelief class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the RasterRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the RasterRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the RasterRelief class as documented in the Conceptual Model.

Abstract Test 168	/ats/Relief/ReliefFeature
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Relief/rc-ReliefFeature
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ReliefFeature class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the ReliefFeature class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ReliefFeature class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ReliefFeature class as documented in the Conceptual Model.

Abstract Test 169	/ats/Relief/TINRelief
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Relief/rc-TINRelief
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TINRelief class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TINRelief class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TINRelief class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TINRelief class as documented in the Conceptual Model.

A.12. Conformance Class Transportation

Abstract Test 170	/ats/Transportation/AbstractTransportationSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-AbstractTransportationSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTransportationSpace abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTransportationSpace abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTransportationSpace abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 171	/ats/Transportation/AuxiliaryTrafficArea
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficArea
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AuxiliaryTrafficArea class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the AuxiliaryTrafficArea class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AuxiliaryTrafficArea class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the AuxiliaryTrafficArea class as documented in the Conceptual Model.

Abstract Test 172	/ats/Transportation/AuxiliaryTrafficAreaClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficAreaClassValue
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficAreaClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 173	/ats/Transportation/AuxiliaryTrafficAreaFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficAreaFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficAreaFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 174	/ats/Transportation/AuxiliaryTrafficAreaUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficAreaUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficAreaUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 175	/ats/Transportation/AuxiliaryTrafficSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AuxiliaryTrafficSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AuxiliaryTrafficSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AuxiliaryTrafficSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the AuxiliaryTrafficSpace class as documented in the Conceptual Model.

Abstract Test 176	/ats/Transportation/AuxiliaryTrafficSpaceClassValue
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficSpaceClassValue
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 177	/ats/Transportation/AuxiliaryTrafficSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 178	/ats/Transportation/AuxiliaryTrafficSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-AuxiliaryTrafficSpaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the AuxiliaryTrafficSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 179	/ats/Transportation/ClearanceSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Transportation/rc-ClearanceSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the ClearanceSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the ClearanceSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the ClearanceSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the ClearanceSpace class as documented in the Conceptual Model.

Abstract Test 180	/ats/Transportation/ClearanceSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-ClearanceSpaceClassValue
Test Method	Manual Inspection
A	Validate that all instances of the ClearanceSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 181	/ats/Transportation/GranularityValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-GranularityValue

Test Method	Manual Inspection
A	Validate that all instances of the GranularityValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 182	/ats/Transportation/Hole
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Hole
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Hole class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Hole class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Hole class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Hole class as documented in the Conceptual Model.

Abstract Test 183	/ats/Transportation/HoleClassName
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-HoleClassName
Test Method	Manual Inspection

A	Validate that all instances of the HoleClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 184	/ats/Transportation/HoleSurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-HoleSurface
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the HoleSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the HoleSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the HoleSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the HoleSurface class as documented in the Conceptual Model.

Abstract Test 185	/ats/Transportation/Intersection
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Intersection
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Intersection class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Intersection class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Intersection class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Intersection class as documented in the Conceptual Model.

Abstract Test 186	/ats/Transportation/IntersectionClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-IntersectionClassValue
Test Method	Manual Inspection
A	Validate that all instances of the IntersectionClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 187	/ats/Transportation/Marking
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Marking
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Marking class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Marking class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Marking class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Marking class as documented in the Conceptual Model.

Abstract Test 188	/ats/Transportation/MarkingClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-MarkingClassValue
Test Method	Manual Inspection
A	Validate that all instances of the MarkingClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 189	/ats/Transportation/Railway
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Railway
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Railway class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Railway class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Railway class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Railway class as documented in the Conceptual Model.

Abstract Test 190	/ats/Transportation/RailwayClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-RailwayClassValue
Test Method	Manual Inspection
A	Validate that all instances of the RailwayClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 191	/ats/Transportation/RailwayFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-RailwayFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the RailwayFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 192	/ats/Transportation/RailwayUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-RailwayUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the RailwayUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 193	/ats/Transportation/Road
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Road
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Road class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Road class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Road class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Road class as documented in the Conceptual Model.

Abstract Test 194	/ats/Transportation/RoadClassValue
--------------------------	--

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-RoadClassValue
Test Method	Manual Inspection
A	Validate that all instances of the RoadClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 195	/ats/Transportation/RoadFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-RoadFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the RoadFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 196	/ats/Transportation/RoadUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-RoadUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the RoadUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 197	/ats/Transportation/Section
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Transportation/rc-Section
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Section class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Section class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Section class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Section class as documented in the Conceptual Model.

Abstract Test 198	/ats/Transportation/SectionClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-SectionClassValue
Test Method	Manual Inspection
A	Validate that all instances of the SectionClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 199	/ats/Transportation/Square
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Square

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Square class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Square class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Square class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Square class as documented in the Conceptual Model.

Abstract Test 200	/ats/Transportation/SquareClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-SquareClassValue
Test Method	Manual Inspection
A	Validate that all instances of the SquareClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 201	/ats/Transportation/SquareFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-SquareFunctionValue
Test Method	Manual Inspection

A	Validate that all instances of the SquareFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--

Abstract Test 202	/ats/Transportation/SquareUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-SquareUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the SquareUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 203	/ats/Transportation/SurfaceMaterialValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-SurfaceMaterialValue
Test Method	Manual Inspection
A	Validate that all instances of the SurfaceMaterialValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 204	/ats/Transportation/Track
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Track
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Track class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Track class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Track class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Track class as documented in the Conceptual Model.

Abstract Test 205	/ats/Transportation/TrackClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrackClassValue
Test Method	Manual Inspection
A	Validate that all instances of the TrackClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 206	/ats/Transportation/TrackFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrackFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TrackFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 207	/ats/Transportation/TrackUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrackUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TrackUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 208	/ats/Transportation/TrafficArea
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficArea
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TrafficArea class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TrafficArea class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TrafficArea class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TrafficArea class as documented in the Conceptual Model.

Abstract Test 209	/ats/Transportation/TrafficAreaClassValue
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficAreaClassValue
Test Method	Manual Inspection
A	Validate that all instances of the TrafficAreaClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 210	/ats/Transportation/TrafficAreaFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficAreaFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TrafficAreaFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 211	/ats/Transportation/TrafficAreaUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficAreaUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TrafficAreaUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 212	/ats/Transportation/TrafficDirectionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	/req/Transportation/rc-TrafficDirectionValue
Test Method	Manual Inspection
A	Validate that all instances of the TrafficDirectionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 213	/ats/Transportation/TrafficSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TrafficSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TrafficSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TrafficSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TrafficSpace class as documented in the Conceptual Model.

Abstract Test 214	/ats/Transportation/TrafficSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficSpaceClassValue

Test Method	Manual Inspection
A	Validate that all instances of the TrafficSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 215	/ats/Transportation/TrafficSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TrafficSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 216	/ats/Transportation/TrafficSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TrafficSpaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TrafficSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 217	/ats/Transportation/TransportationSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TransportationSpaceClassValue
Test Method	Manual Inspection

A	Validate that all instances of the TransportationSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	--

Abstract Test 218	/ats/Transportation/TransportationSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TransportationSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TransportationSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 219	/ats/Transportation/TransportationSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-TransportationSpaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TransportationSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 220	/ats/Transportation/Waterway
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Transportation/rc-Waterway
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Waterway class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the Waterway class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Waterway class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Waterway class as documented in the Conceptual Model.

Abstract Test 221	/ats/Transportation/WaterwayClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-WaterwayClassValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterwayClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 222	/ats/Transportation/WaterwayFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-WaterwayFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterwayFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 223	/ats/Transportation/WaterwayUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Transportation/rc-WaterwayUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterwayUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

A.13. Conformance Class Tunnel

Abstract Test 224	/ats/Tunnel/AbstractTunnel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-AbstractTunnel
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractTunnel abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractTunnel abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractTunnel abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 225	/ats/Tunnel/HollowSpace
--------------------------	-------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-HollowSpace
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the HollowSpace class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the HollowSpace class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the HollowSpace class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the HollowSpace class as documented in the Conceptual Model.

Abstract Test 226	/ats/Tunnel/HollowSpaceClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-HollowSpaceClassValue
Test Method	Manual Inspection
A	Validate that all instances of the HollowSpaceClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 227	/ats/Tunnel/HollowSpaceFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	/req/Tunnel/rc-HollowSpaceFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the HollowSpaceFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 228	/ats/Tunnel/HollowSpaceUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-HollowSpaceUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the HollowSpaceUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 229	/ats/Tunnel/Tunnel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-Tunnel
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Tunnel class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Tunnel class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the Tunnel class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Tunnel class as documented in the Conceptual Model.

Abstract Test 230	/ats/Tunnel/TunnelClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelClassValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 231	/ats/Tunnel/TunnelConstructiveElement
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelConstructiveElement
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelConstructiveElement class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TunnelConstructiveElement class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TunnelConstructiveElement class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelConstructiveElement class as documented in the Conceptual Model.

Abstract Test 232	/ats/Tunnel/TunnelConstructiveElementClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelConstructiveElementClassValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelConstructiveElementClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 233	/ats/Tunnel/TunnelConstructiveElementFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelConstructiveElementFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelConstructiveElementFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 234	/ats/Tunnel/TunnelConstructiveElementUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	/req/Tunnel/rc-TunnelConstructiveElementUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelConstructiveElementUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 235	/ats/Tunnel/TunnelFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 236	/ats/Tunnel/TunnelFurniture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelFurniture
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelFurniture class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TunnelFurniture class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.

C	Validate that the data element has the same properties (attributes) as those specified for the TunnelFurniture class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelFurniture class as documented in the Conceptual Model.

Abstract Test 237	/ats/Tunnel/TunnelFurnitureClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelFurnitureClassValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelFurnitureClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 238	/ats/Tunnel/TunnelFurnitureFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelFurnitureFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelFurnitureFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 239	/ats/Tunnel/TunnelFurnitureUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelFurnitureUsageValue

Test Method	Manual Inspection
A	Validate that all instances of the TunnelFurnitureUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 240	/ats/Tunnel/TunnelInstallation
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelInstallation
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelInstallation class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the TunnelInstallation class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TunnelInstallation class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelInstallation class as documented in the Conceptual Model.

Abstract Test 241	/ats/Tunnel/TunnelInstallationClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelInstallationClassValue
Test Method	Manual Inspection

A	Validate that all instances of the TunnelInstallationClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model
---	---

Abstract Test 242	/ats/Tunnel/TunnelInstallationFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelInstallationFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelInstallationFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 243	/ats/Tunnel/TunnelInstallationUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelInstallationUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelInstallationUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 244	/ats/Tunnel/TunnelPart
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelPart
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the TunnelPart class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the TunnelPart class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the TunnelPart class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the TunnelPart class as documented in the Conceptual Model.

Abstract Test 245	/ats/Tunnel/TunnelUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Tunnel/rc-TunnelUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the TunnelUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

A.14. Conformance Class Vegetation

Abstract Test 246	/ats/Vegetation/AbstractVegetationObject
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Vegetation/rc-AbstractVegetationObject
Test Method	Manual Inspection

A	Validate that a date element exists with the same definition as that of the AbstractVegetationObject abstract class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the AbstractVegetationObject abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractVegetationObject abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 247	/ats/Vegetation/PlantCover
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Vegetation/rc-PlantCover
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the PlantCover class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the PlantCover class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the PlantCover class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the PlantCover class as documented in the Conceptual Model.

Abstract Test 248	/ats/Vegetation/PlantCoverClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-PlantCoverClassValue
Test Method	Manual Inspection
A	Validate that all instances of the PlantCoverClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 249	/ats/Vegetation/PlantCoverFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-PlantCoverFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the PlantCoverFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 250	/ats/Vegetation/PlantCoverUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-PlantCoverUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the PlantCoverUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 251	/ats/Vegetation/SolitaryVegetationObject
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Vegetation/rc-SolitaryVegetationObject
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the SolitaryVegetationObject class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the SolitaryVegetationObject class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the SolitaryVegetationObject class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the SolitaryVegetationObject class as documented in the Conceptual Model.

Abstract Test 252	/ats/Vegetation/SolitaryVegetationObjectClassValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-SolitaryVegetationObjectClassValue
Test Method	Manual Inspection
A	Validate that all instances of the SolitaryVegetationObjectClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 253	/ats/Vegetation/SolitaryVegetationObjectFunctionValue
--------------------------	--

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-SolitaryVegetationObjectFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the SolitaryVegetationObjectFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 254	/ats/Vegetation/SolitaryVegetationObjectUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-SolitaryVegetationObjectUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the SolitaryVegetationObjectUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 255	/ats/Vegetation/SpeciesValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Vegetation/rc-SpeciesValue
Test Method	Manual Inspection
A	Validate that all instances of the SpeciesValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

A.15. Conformance Class Versioning

Abstract Test 256	/ats/Versioning/Transaction
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	/req/Versioning/rc-Transaction
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Transaction type in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Transaction type in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Transaction type in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Transaction type as documented in the Conceptual Model.

Abstract Test 257	/ats/Versioning/TransactionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Versioning/rc-TransactionValue
Test Method	Manual Inspection
A	Validate that all instances of the TransactionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 258	/ats/Versioning/TransitionValue
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Versioning/rc-TransitionValue
Test Method	Manual Inspection
A	Validate that all instances of the TransitionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 259	/ats/Versioning/Version
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Versioning/rc-Version
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the Version class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the Version class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the Version class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the Version class as documented in the Conceptual Model.

Abstract Test 260	/ats/Versioning/VersionTransition
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Versioning/rc-VersionTransition
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the VersionTransition class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the VersionTransition class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the VersionTransition class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the VersionTransition class as documented in the Conceptual Model.

A.16. Conformance Class WaterBody

Abstract Test 261	/ats/Waterbody/AbstractWaterBoundarySurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	/req/Waterbody/rc-AbstractWaterBoundarySurface
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the AbstractWaterBoundarySurface abstract class in the Conceptual Model

B	Validate that the data element has the same relationships with other elements as those defined for the AbstractWaterBoundarySurface abstract class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the AbstractWaterBoundarySurface abstract class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.

Abstract Test 262	/ats/Waterbody/WaterBody
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Waterbody/rc-WaterBody
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WaterBody class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WaterBody class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WaterBody class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the WaterBody class as documented in the Conceptual Model.

Abstract Test 263	/ats/Waterbody/WaterBodyClassValue
--------------------------	------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Waterbody/rc-WaterBodyClassValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterBodyClassValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 264	/ats/Waterbody/WaterBodyFunctionValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Waterbody/rc-WaterBodyFunctionValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterBodyFunctionValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 265	/ats/Waterbody/WaterBodyUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Waterbody/rc-WaterBodyUsageValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterBodyUsageValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 266	/ats/Waterbody/WaterGroundSurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	/req/Waterbody/rc-WaterGroundSurface
Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WaterGroundSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WaterGroundSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WaterGroundSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the WaterGroundSurface class as documented in the Conceptual Model.

Abstract Test 267	/ats/Waterbody/WaterLevelValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	/req/Waterbody/rc-WaterLevelValue
Test Method	Manual Inspection
A	Validate that all instances of the WaterLevelValue codelist in the Implementation Specification can only take the values specified for that codelist in the Conceptual Model

Abstract Test 268	/ats/Waterbody/WaterSurface
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	/req/Waterbody/rc-WaterSurface

Test Method	Manual Inspection
A	Validate that a date element exists with the same definition as that of the WaterSurface class in the Conceptual Model
B	Validate that the data element has the same relationships with other elements as those defined for the WaterSurface class in the Conceptual Model. Validate that those relationships have the same source, target, direction, and cardinalities as those documented in the Conceptual Model.
C	Validate that the data element has the same properties (attributes) as those specified for the WaterSurface class in the Conceptual Model. Validate that those properties have the same name, definition, type, and cardinality of those documented in the Conceptual Model.
D	Validate that the properties of the data element include those of all parent classes of the WaterSurface class as documented in the Conceptual Model.

Annex B: Title ({Normative/Informative})

NOTE

Place other Annex material in sequential annexes beginning with "B" and leave final two annexes for the Revision History and Bibliography

Annex C: Revision History

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

Chapter 14. Changelog for CityGML 3.0

The following table lists all feature types, properties, and data types which have been added or changed for CityGML 3.0.

Feature Class / Data Type	Property	New	Changed	Deleted	Description of Change

Annex D: Bibliography

Example Bibliography (Delete this note).

The TC has approved Springer LNCS as the official document citation type.

Springer LNCS is widely used in technical and computer science journals and other publications

NOTE

- For citations in the text please use square brackets and consecutive numbers:
[1], [2], [3]

– Actual References:

[n] Journal: Author Surname, A.: Title. Publication Title. Volume number, Issue number, Pages Used (Year Published)

[n] Web: Author Surname, A.: Title, <http://Website-Url>

[1] OGC: OGC Testbed 12 Annex B: Architecture. (2015).