

# **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. <b>term name</b> .....	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
11.1. Appearance Model .....	62
11.1.1. <b>Apearance Package</b> .....	63
11.1.2. <b>Class AbstractSurfaceData</b> .....	64
11.1.3. <b>Class AbstractTexture</b> .....	64
11.1.4. <b>Class Appearance</b> .....	65
11.1.5. <b>Class Color</b> .....	66
11.1.6. <b>Class ColorPlusOpacity</b> .....	67
11.1.7. <b>Class GeoreferencedTexture</b> .....	68
11.1.8. <b>Class ParameterizedTexture</b> .....	69
11.1.9. <b>Class TextureAssociation</b> .....	70
11.1.10. <b>Class X3DMaterial</b> .....	71
11.1.11. <b>Class AbstractTextureParameterization</b> .....	72
11.1.12. <b>Class TexCoordGen</b> .....	73
11.1.13. <b>Class TexCoordList</b> .....	74
11.1.14. <b>Class TextureType</b> .....	75
11.1.15. <b>Class WrapMode</b> .....	76
11.1.16. Additional Information .....	77
11.2. Core .....	77
11.2.1. <b>Core Package</b> .....	78
11.2.2. <b>Class AbstractAppearance</b> .....	79
11.2.3. <b>Class AbstractCityObject</b> .....	79
11.2.4. <b>Class AbstractDynamizer</b> .....	81
11.2.5. <b>Class AbstractFeatureWithLifespan</b> .....	81
11.2.6. <b>Class AbstractLogicalSpace</b> .....	82
11.2.7. <b>Class AbstractOccupiedSpace</b> .....	83
11.2.8. <b>Class AbstractPhysicalSpace</b> .....	84
11.2.9. <b>Class AbstractPointCloud</b> .....	85
11.2.10. <b>Class AbstractSpace</b> .....	85
11.2.11. <b>Class AbstractSpaceBoundary</b> .....	88
11.2.12. <b>Class AbstractThematicSurface</b> .....	88
11.2.13. <b>Class AbstractUnoccupiedSpace</b> .....	89
11.2.14. <b>Class AbstractVersion</b> .....	90
11.2.15. <b>Class AbstractVersionTransition</b> .....	91
11.2.16. <b>Class Address</b> .....	91
11.2.17. <b>Class CityModel</b> .....	92
11.2.18. <b>Class CityObjectRelation</b> .....	93
11.2.19. <b>Class ClosureSurface</b> .....	94
11.2.20. <b>Class DoubleBetween0and1</b> .....	95
11.2.21. <b>Class DoubleBetween0and1List</b> .....	96

<b>11.2.22. Class DoubleList</b>	96
<b>11.2.23. Class ImplicitGeometry</b>	97
<b>11.2.24. Class IntegerBetween0and3</b>	99
<b>11.2.25. Class IntervalValue</b>	99
<b>11.2.26. Class MimeValue</b>	100
<b>11.2.27. Class OccupantTypeValue</b>	100
<b>11.2.28. Class OtherRelationTypeValue</b>	100
<b>11.2.29. Class QualifiedAreaValue</b>	101
<b>11.2.30. Class QualifiedVolumeValue</b>	101
<b>11.2.31. Class RelationTypeValue</b>	102
<b>11.2.32. Class TemporalRelationTypeValue</b>	102
<b>11.2.33. Class TopologicRelationTypeValue</b>	102
<b>11.2.34. Class TransformationMatrix2x2</b>	103
<b>11.2.35. Class TransformationMatrix3x4</b>	103
<b>11.2.36. Class TransformationMatrix4x4</b>	104
<b>11.2.37. Class AbstractGenericAttribute</b>	105
<b>11.2.38. Class ExternalReference</b>	106
<b>11.2.39. Class Occupancy</b>	107
<b>11.2.40. Class QualifiedArea</b>	108
<b>11.2.41. Class QualifiedVolume</b>	108
<b>11.2.42. Class RelativeToTerrain</b>	109
<b>11.2.43. Class RelativeToWater</b>	110
<b>11.2.44. Class SpaceType</b>	111
<b>11.2.45. Class XALAddressDetails</b>	112
<b>11.2.46. Additional Information</b>	113
<b>11.3. Digital Terrain Model</b>	113
<b>11.3.1. Relief Package</b>	114
<b>11.3.2. Class AbstractReliefComponent</b>	114
<b>11.3.3. Class BreaklineRelief</b>	115
<b>11.3.4. Class MassPointRelief</b>	116
<b>11.3.5. Class RasterRelief</b>	117
<b>11.3.6. Class ReliefFeature</b>	118
<b>11.3.7. Class TINRelief</b>	119
<b>11.3.8. Additional Information</b>	119
<b>11.4. Building Model</b>	120
<b>11.4.1. Building Package</b>	121
<b>11.4.2. Class AbstractBuilding</b>	121
<b>11.4.3. Class AbstractBuildingSubdivision</b>	123
<b>11.4.4. Class Building</b>	125
<b>11.4.5. Class BuildingClassValue</b>	125
<b>11.4.6. Class BuildingConstructiveElement</b>	126

<b>11.4.7. Class BuildingConstructiveElementClassValue</b>	127
<b>11.4.8. Class BuildingConstructiveElementFunctionValue</b>	127
<b>11.4.9. Class BuildingConstructiveElementUsageValue</b>	128
<b>11.4.10. Class BuildingFunctionValue</b>	128
<b>11.4.11. Class BuildingFurniture</b>	128
<b>11.4.12. Class BuildingFurnitureClassValue</b>	130
<b>11.4.13. Class BuildingFurnitureFunctionValue</b>	130
<b>11.4.14. Class BuildingFurnitureUsageValue</b>	130
<b>11.4.15. Class BuildingInstallation</b>	131
<b>11.4.16. Class BuildingInstallationClassValue</b>	132
<b>11.4.17. Class BuildingInstallationFunctionValue</b>	132
<b>11.4.18. Class BuildingInstallationUsageValue</b>	133
<b>11.4.19. Class BuildingPart</b>	133
<b>11.4.20. Class BuildingRoom</b>	134
<b>11.4.21. Class BuildingRoomClassValue</b>	135
<b>11.4.22. Class BuildingRoomFunctionValue</b>	135
<b>11.4.23. Class BuildingRoomUsageValue</b>	136
<b>11.4.24. Class BuildingSubdivisionClassValue</b>	136
<b>11.4.25. Class BuildingSubdivisionFunctionValue</b>	137
<b>11.4.26. Class BuildingSubdivisionUsageValue</b>	137
<b>11.4.27. Class BuildingUnit</b>	137
<b>11.4.28. Class BuildingUsageValue</b>	138
<b>11.4.29. Class RoofTypeValue</b>	138
<b>11.4.30. Class RoomElevationReferenceValue</b>	139
<b>11.4.31. Class Storey</b>	139
<b>11.4.32. Class RoomHeight</b>	140
<b>11.4.33. Additional Information</b>	141
<b>11.4.34. Boundary surfaces</b>	143
<b>11.4.35. Openings</b>	145
<b>11.4.36. Building Interior</b>	145
<b>11.4.37. Modelling building storeys using CityObjectGroups</b>	146
<b>11.5. Tunnel</b>	146
<b>11.5.1. Tunnel Package</b>	147
<b>11.5.2. Class AbstractTunnel</b>	147
<b>11.5.3. Class HollowSpace</b>	149
<b>11.5.4. Class HollowSpaceClassValue</b>	150
<b>11.5.5. Class HollowSpaceFunctionValue</b>	151
<b>11.5.6. Class HollowSpaceUsageValue</b>	151
<b>11.5.7. Class Tunnel</b>	151
<b>11.5.8. Class TunnelClassValue</b>	152
<b>11.5.9. Class TunnelConstructiveElement</b>	152

<b>11.5.10. Class TunnelConstructiveElementClassValue</b>	153
<b>11.5.11. Class TunnelConstructiveElementFunctionValue</b>	154
<b>11.5.12. Class TunnelConstructiveElementUsageValue</b>	154
<b>11.5.13. Class TunnelFunctionValue</b>	155
<b>11.5.14. Class TunnelFurniture</b>	155
<b>11.5.15. Class TunnelFurnitureClassValue</b>	156
<b>11.5.16. Class TunnelFurnitureFunctionValue</b>	157
<b>11.5.17. Class TunnelFurnitureUsageValue</b>	157
<b>11.5.18. Class TunnelInstallation</b>	157
<b>11.5.19. Class TunnelInstallationClassValue</b>	158
<b>11.5.20. Class TunnelInstallationFunctionValue</b>	159
<b>11.5.21. Class TunnelInstallationUsageValue</b>	159
<b>11.5.22. Class TunnelPart</b>	160
<b>11.5.23. Class TunnelUsageValue</b>	160
<b>11.5.24. Additional Information</b>	161
<b>11.6. Bridge Model</b>	161
<b>  11.6.1. Bridge Package</b>	162
<b>  11.6.2. Class AbstractBridge</b>	162
<b>  11.6.3. Class Bridge</b>	164
<b>  11.6.4. Class BridgeClassValue</b>	165
<b>  11.6.5. Class BridgeConstructiveElement</b>	165
<b>  11.6.6. Class BridgeConstructiveElementClassValue</b>	166
<b>  11.6.7. Class BridgeConstructiveElementFunctionValue</b>	167
<b>  11.6.8. Class BridgeConstructiveElementUsageValue</b>	167
<b>  11.6.9. Class BridgeFunctionValue</b>	167
<b>  11.6.10. Class BridgeFurniture</b>	168
<b>  11.6.11. Class BridgeFurnitureClassValue</b>	169
<b>  11.6.12. Class BridgeFurnitureFunctionValue</b>	169
<b>  11.6.13. Class BridgeFurnitureUsageValue</b>	170
<b>  11.6.14. Class BridgeInstallation</b>	170
<b>  11.6.15. Class BridgeInstallationClassValue</b>	171
<b>  11.6.16. Class BridgeInstallationFunctionValue</b>	172
<b>  11.6.17. Class BridgeInstallationUsageValue</b>	172
<b>  11.6.18. Class BridgePart</b>	172
<b>  11.6.19. Class BridgeRoom</b>	173
<b>  11.6.20. Class BridgeRoomClassValue</b>	174
<b>  11.6.21. Class BridgeRoomFunctionValue</b>	175
<b>  11.6.22. Class BridgeRoomUsageValue</b>	175
<b>  11.6.23. Class BridgeUsageValue</b>	175
<b>  11.6.24. Additional Information</b>	176
<b>11.7. Water Body</b>	176

<b>11.7.1. WaterBody Package</b>	177
<b>11.7.2. Class AbstractWaterBoundarySurface</b>	177
<b>11.7.3. Class WaterBody</b>	178
<b>11.7.4. Class WaterBodyClassValue</b>	179
<b>11.7.5. Class WaterBodyFunctionValue</b>	180
<b>11.7.6. Class WaterBodyUsageValue</b>	180
<b>11.7.7. Class WaterGroundSurface</b>	180
<b>11.7.8. Class WaterLevelValue</b>	181
<b>11.7.9. Class WaterSurface</b>	181
<b>11.7.10. Additional Information</b>	182
<b>11.8. Transportation</b>	182
<b>  11.8.1. Transportation Package</b>	183
<b>  11.8.2. Class AbstractTransportationSpace</b>	183
<b>  11.8.3. Class AuxiliaryTrafficArea</b>	185
<b>  11.8.4. Class AuxiliaryTrafficAreaClassValue</b>	186
<b>  11.8.5. Class AuxiliaryTrafficAreaFunctionValue</b>	186
<b>  11.8.6. Class AuxiliaryTrafficAreaUsageValue</b>	187
<b>  11.8.7. Class AuxiliaryTrafficSpace</b>	187
<b>  11.8.8. Class AuxiliaryTrafficSpaceClassValue</b>	189
<b>  11.8.9. Class AuxiliaryTrafficSpaceFunctionValue</b>	189
<b>  11.8.10. Class AuxiliaryTrafficSpaceUsageValue</b>	189
<b>  11.8.11. Class ClearanceSpace</b>	190
<b>  11.8.12. Class ClearanceSpaceClassValue</b>	191
<b>  11.8.13. Class Hole</b>	191
<b>  11.8.14. Class HoleClassValue</b>	192
<b>  11.8.15. Class HoleSurface</b>	192
<b>  11.8.16. Class Intersection</b>	193
<b>  11.8.17. Class IntersectionClassValue</b>	194
<b>  11.8.18. Class Marking</b>	194
<b>  11.8.19. Class MarkingClassValue</b>	195
<b>  11.8.20. Class Railway</b>	195
<b>  11.8.21. Class RailwayClassValue</b>	196
<b>  11.8.22. Class RailwayFunctionValue</b>	197
<b>  11.8.23. Class RailwayUsageValue</b>	197
<b>  11.8.24. Class Road</b>	197
<b>  11.8.25. Class RoadClassValue</b>	199
<b>  11.8.26. Class RoadFunctionValue</b>	199
<b>  11.8.27. Class RoadUsageValue</b>	199
<b>  11.8.28. Class Section</b>	200
<b>  11.8.29. Class SectionClassValue</b>	200
<b>  11.8.30. Class Square</b>	201

<b>11.8.31. Class SquareClassValue</b>	202
<b>11.8.32. Class SquareFunctionValue</b>	202
<b>11.8.33. Class SquareUsageValue</b>	202
<b>11.8.34. Class SurfaceMaterialValue</b>	203
<b>11.8.35. Class Track</b>	203
<b>11.8.36. Class TrackClassValue</b>	204
<b>11.8.37. Class TrackFunctionValue</b>	205
<b>11.8.38. Class TrackUsageValue</b>	205
<b>11.8.39. Class TrafficArea</b>	205
<b>11.8.40. Class TrafficAreaClassValue</b>	207
<b>11.8.41. Class TrafficAreaFunctionValue</b>	207
<b>11.8.42. Class TrafficAreaUsageValue</b>	207
<b>11.8.43. Class TrafficSpace</b>	208
<b>11.8.44. Class TrafficSpaceClassValue</b>	210
<b>11.8.45. Class TrafficSpaceFunctionValue</b>	210
<b>11.8.46. Class TrafficSpaceUsageValue</b>	210
<b>11.8.47. Class TransportationSpaceClassValue</b>	211
<b>11.8.48. Class TransportationSpaceFunctionValue</b>	211
<b>11.8.49. Class TransportationSpaceUsageValue</b>	211
<b>11.8.50. Class Waterway</b>	212
<b>11.8.51. Class WaterwayClassValue</b>	213
<b>11.8.52. Class WaterwayFunctionValue</b>	213
<b>11.8.53. Class WaterwayUsageValue</b>	214
<b>11.8.54. Class GranularityValue</b>	214
<b>11.8.55. Class TrafficDirectionValue</b>	215
<b>11.8.56. Additional Information</b>	216
<b>11.9. Vegetation</b>	216
<b>  11.9.1. Vegetation Package</b>	217
<b>  11.9.2. Class AbstractVegetationObject</b>	218
<b>  11.9.3. Class PlantCover</b>	218
<b>  11.9.4. Class PlantCoverClassValue</b>	220
<b>  11.9.5. Class PlantCoverFunctionValue</b>	220
<b>  11.9.6. Class PlantCoverUsageValue</b>	220
<b>  11.9.7. Class SolitaryVegetationObject</b>	221
<b>  11.9.8. Class SolitaryVegetationObjectClassValue</b>	223
<b>  11.9.9. Class SolitaryVegetationObjectFunctionValue</b>	223
<b>  11.9.10. Class SolitaryVegetationObjectUsageValue</b>	223
<b>  11.9.11. Class SpeciesValue</b>	224
<b>  11.9.12. Additional Information</b>	224
<b>11.10. City Furniture Model</b>	224
<b>  11.10.1. CityFurniture Package</b>	225

<b>11.10.2. Class CityFurniture</b>	225
<b>11.10.3. Class CityFurnitureClassValue</b>	227
<b>11.10.4. Class CityFurnitureFunctionValue</b>	227
<b>11.10.5. Class CityFurnitureUsageValue</b>	227
<b>11.10.6. Additional Information</b>	228
<b>11.11. Land Use</b>	228
<b>  11.11.1. LandUse Package</b>	229
<b>  11.11.2. Class LandUse</b>	229
<b>  11.11.3. Class LandUseClassValue</b>	230
<b>  11.11.4. Class LandUseFunctionValue</b>	231
<b>  11.11.5. Class LandUseUsageValue</b>	231
<b>  11.11.6. Additional Information</b>	232
<b>11.12. City Object Group</b>	232
<b>  11.12.1. CityObjectGroup Package</b>	233
<b>  11.12.2. Class CityObjectGroup</b>	233
<b>  11.12.3. Class CityObjectGroupClassValue</b>	234
<b>  11.12.4. Class CityObjectGroupFunctionValue</b>	235
<b>  11.12.5. Class CityObjectGroupUsageValue</b>	235
<b>  11.12.6. Class Role</b>	235
<b>  11.12.7. Additional Information</b>	236
<b>11.13. Generics</b>	236
<b>  11.13.1. Generics Package</b>	237
<b>  11.13.2. Class GenericLogicalSpace</b>	237
<b>  11.13.3. Class GenericLogicalSpaceClassValue</b>	238
<b>  11.13.4. Class GenericLogicalSpaceFunctionValue</b>	239
<b>  11.13.5. Class GenericLogicalSpaceUsageValue</b>	239
<b>  11.13.6. Class GenericOccupiedSpace</b>	240
<b>  11.13.7. Class GenericOccupiedSpaceClassValue</b>	241
<b>  11.13.8. Class GenericOccupiedSpaceFunctionValue</b>	241
<b>  11.13.9. Class GenericOccupiedSpaceUsageValue</b>	241
<b>  11.13.10. Class GenericThematicSurface</b>	242
<b>  11.13.11. Class GenericThematicSurfaceClassValue</b>	243
<b>  11.13.12. Class GenericThematicSurfaceFunctionValue</b>	243
<b>  11.13.13. Class GenericThematicSurfaceUsageValue</b>	244
<b>  11.13.14. Class GenericUnoccupiedSpace</b>	244
<b>  11.13.15. Class GenericUnoccupiedSpaceClassValue</b>	245
<b>  11.13.16. Class GenericUnoccupiedSpaceFunctionValue</b>	245
<b>  11.13.17. Class GenericUnoccupiedSpaceUsageValue</b>	246
<b>  11.13.18. Class DateAttribute</b>	246
<b>  11.13.19. Class DoubleAttribute</b>	247
<b>  11.13.20. Class GenericAttributeSet</b>	248

<b>11.13.21. Class IntAttribute</b>	249
<b>11.13.22. Class MeasureAttribute</b>	250
<b>11.13.23. Class StringAttribute</b>	251
<b>11.13.24. Class UriAttribute</b>	252
<b>11.13.25. Additional Information</b>	253
<b>11.14. Construction</b>	253
<b>  11.14.1. Construction Package</b>	254
<b>  11.14.2. Class AbstractConstruction</b>	254
<b>  11.14.3. Class AbstractConstructionSurface</b>	256
<b>  11.14.4. Class AbstractConstructiveElement</b>	257
<b>  11.14.5. Class AbstractFillingElement</b>	258
<b>  11.14.6. Class AbstractFillingSurface</b>	258
<b>  11.14.7. Class AbstractFurniture</b>	259
<b>  11.14.8. Class AbstractInstallation</b>	260
<b>  11.14.9. Class CeilingSurface</b>	261
<b>  11.14.10. Class Door</b>	261
<b>  11.14.11. Class DoorClassValue</b>	262
<b>  11.14.12. Class DoorFunctionValue</b>	263
<b>  11.14.13. Class DoorSurface</b>	263
<b>  11.14.14. Class DoorUsageValue</b>	264
<b>  11.14.15. Class ElevationReferenceValue</b>	264
<b>  11.14.16. Class EventValue</b>	265
<b>  11.14.17. Class FloorSurface</b>	265
<b>  11.14.18. Class GroundSurface</b>	266
<b>  11.14.19. Class InteriorWallSurface</b>	266
<b>  11.14.20. Class OtherConstruction</b>	267
<b>  11.14.21. Class OtherConstructionClassValue</b>	268
<b>  11.14.22. Class OtherConstructionFunctionValue</b>	268
<b>  11.14.23. Class OtherConstructionUsageValue</b>	269
<b>  11.14.24. Class OuterCeilingSurface</b>	269
<b>  11.14.25. Class OuterFloorSurface</b>	270
<b>  11.14.26. Class RoofSurface</b>	270
<b>  11.14.27. Class WallSurface</b>	271
<b>  11.14.28. Class Window</b>	272
<b>  11.14.29. Class WindowClassValue</b>	273
<b>  11.14.30. Class WindowFunctionValue</b>	273
<b>  11.14.31. Class WindowSurface</b>	274
<b>  11.14.32. Class WindowUsageValue</b>	274
<b>  11.14.33. Class ConditionOfConstructionValue</b>	275
<b>  11.14.34. Class ConstructionEvent</b>	276
<b>  11.14.35. Class Elevation</b>	277

11.14.36. Class Height .....	278
11.14.37. Class HeightStatusValue .....	279
11.14.38. Class RelationToConstruction .....	279
11.14.39. Additional Information .....	280
11.15. Dynamizer .....	280
11.15.1. Dynamizer Package .....	281
11.15.2. Class AbstractAtomicTimeseries .....	281
11.15.3. Class AbstractTimeseries .....	282
11.15.4. Class AuthenticationValue .....	283
11.15.5. Class CompositeTimeseries .....	284
11.15.6. Class Dynamizer .....	284
11.15.7. Class GenericTimeseries .....	285
11.15.8. Class SensorConnectionValue .....	286
11.15.9. Class StandardFileTimeseries .....	287
11.15.10. Class StandardFieldValue .....	288
11.15.11. Class TabulatedFileTimeseries .....	288
11.15.12. Class TabulatedFieldValue .....	290
11.15.13. Class SensorConnection .....	291
11.15.14. Class TimeseriesComponent .....	293
11.15.15. Class TimeseriesValue .....	294
11.15.16. Class TimeValuePair .....	295
11.15.17. Additional Information .....	297
12. Media Types for any data encoding(s) .....	298
Annex A: Conformance Class Abstract Test Suite (Normative) .....	299
A.1. Conformance Class Appearance .....	299
A.2. Conformance Class Bridge .....	308
A.3. Conformance Class Building .....	317
A.4. Conformance Class CityFurniture .....	331
A.5. Conformance Class CityObjectGroup .....	332
A.6. Conformance Class Core .....	335
A.7. Conformance Class Dynamizer .....	359
A.8. Conformance Class Generics .....	367
A.9. Conformance Class LandUse .....	378
A.10. Conformance Class PointCloud .....	380
A.11. Conformance Class Relief .....	381
A.12. Conformance Class Transportation .....	385
A.13. Conformance Class Tunnel .....	407
A.14. Conformance Class Vegetation .....	416
A.15. Conformance Class Versioning .....	420
A.16. Conformance Class WaterBody .....	423
Annex B: Title ( {Normative/Informative} ) .....	428

Annex C: Revision History .....	429
13. Changelog for CityGML 3.0 .....	430
Annex D: Bibliography .....	431

## **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.

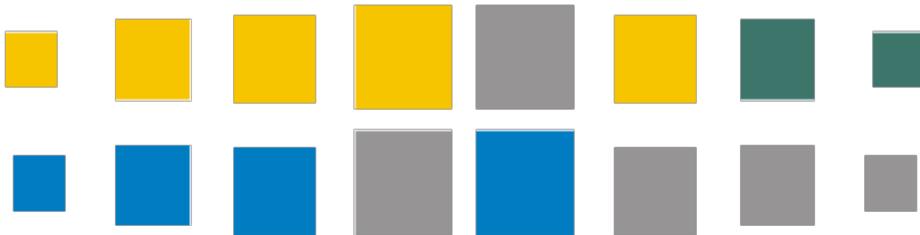
Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

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>

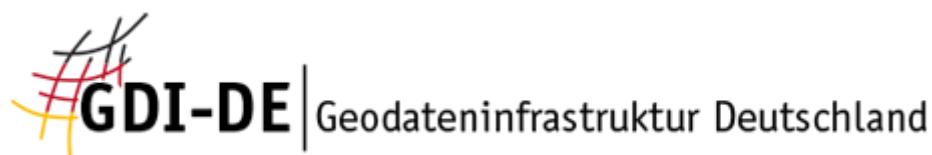
**NOTE**

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](http://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	<a href="mailto:thomas.kolbe@tu-berlin.de">thomas.kolbe@tu-berlin.de</a> <a href="mailto:claus.nagel@tu-berlin.de">claus.nagel@tu-berlin.de</a> <a href="mailto:alexandra.lorenz@tu-berlin.de">alexandra.lorenz@tu-berlin.de</a>
Dr. Gerhard Gröger, Prof. Dr. Lutz Plümer, Angela Czerwinski	Institute for Geodesy and Geoinformation, University of Bonn	<a href="mailto:Groeger@ikg.uni-bonn.de">Groeger@ikg.uni-bonn.de</a> <a href="mailto:Pluemer@ikg.uni-bonn.de">Pluemer@ikg.uni-bonn.de</a> <a href="mailto:Czerwinski@ikg.uni-bonn.de">Czerwinski@ikg.uni-bonn.de</a>
Haik Lorenz	Autodesk, Inc.	<a href="mailto:haik.lorenz@autodesk.com">haik.lorenz@autodesk.com</a>
Alain Lapierre, Stefan Apfel, Paul Scarponcini	Bentley Systems, Inc.	<a href="mailto:alain.lapierre@bentley.com">alain.lapierre@bentley.com</a> <a href="mailto:stefan.apfel@bentley.com">stefan.apfel@bentley.com</a> <a href="mailto:paul.scarponcini@bentley.com">paul.scarponcini@bentley.com</a>
Carsten Rönsdorf	Ordnance Survey, Great Britain	<a href="mailto:carsten.roensdorf@ordnancesurvey.co.uk">carsten.roensdorf@ordnancesurvey.co.uk</a>

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

## 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

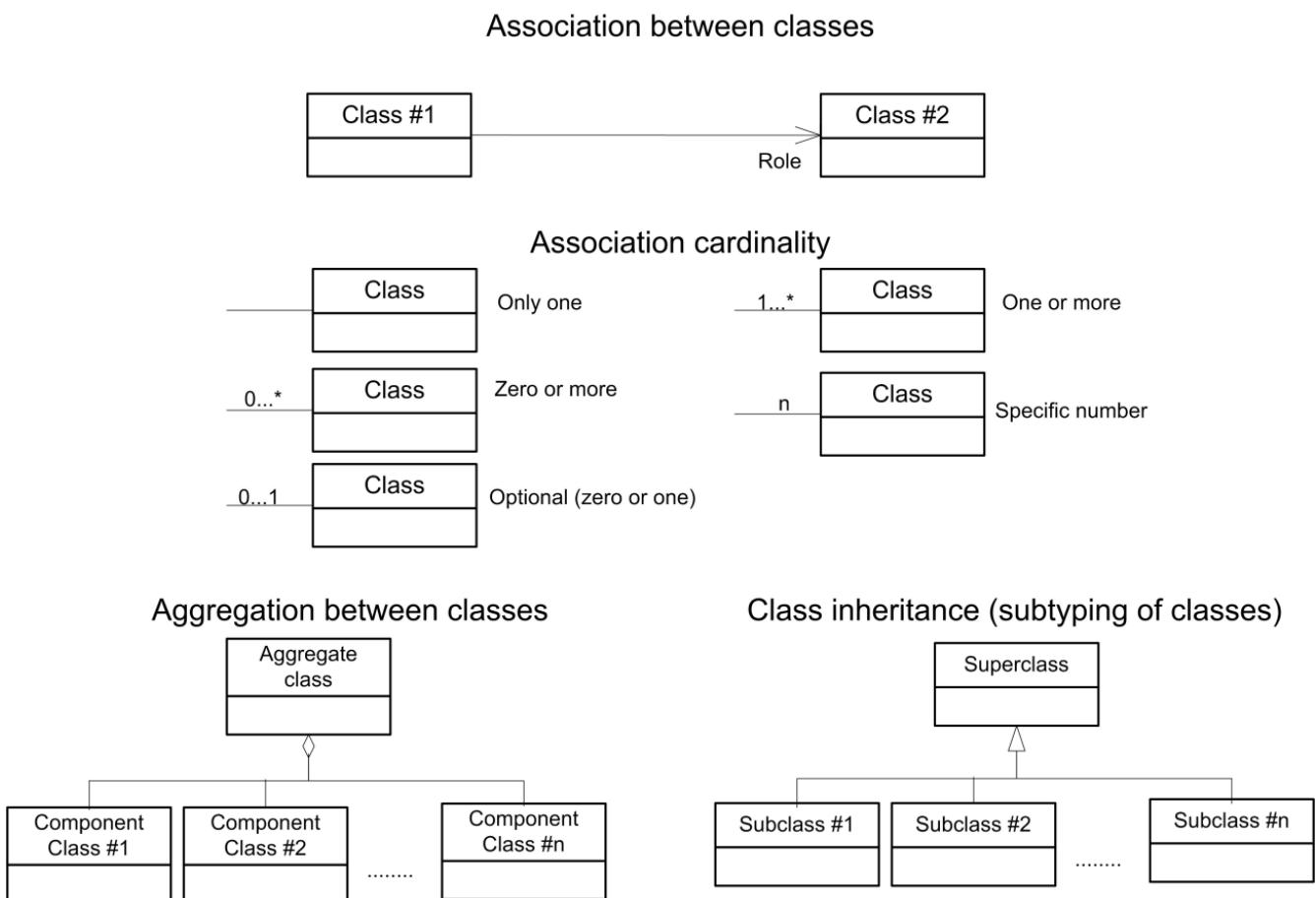
The normative provisions in this document are denoted by the URI

<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)

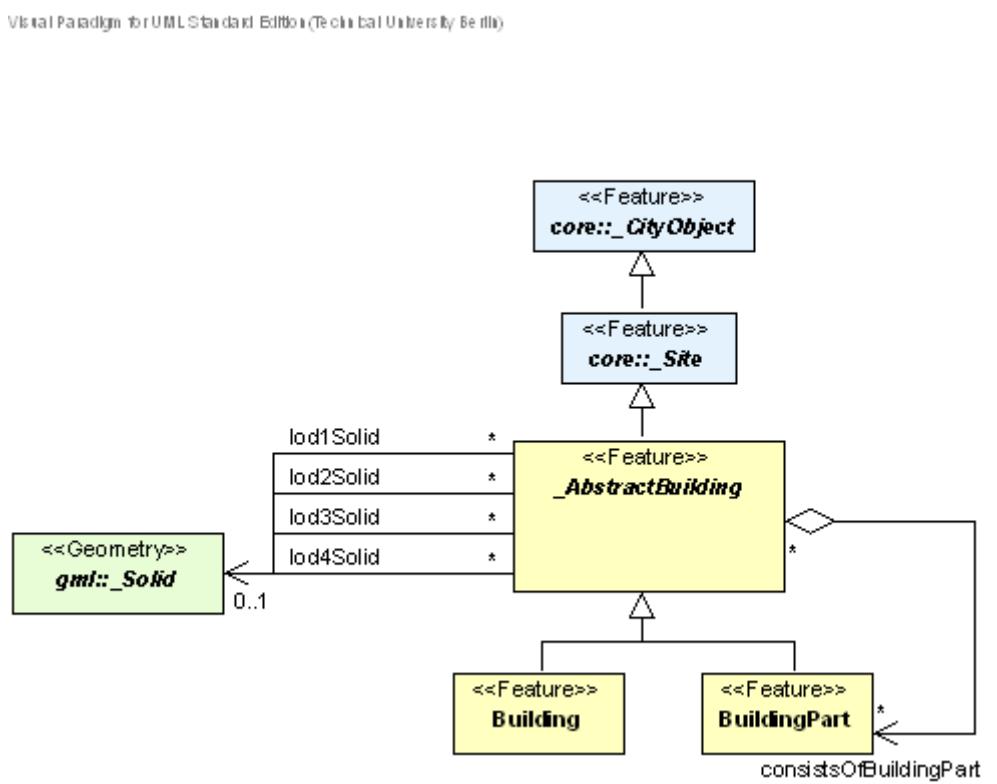


Figure 2. Example UML diagram demonstrating the UML notation and coloring scheme used throughout the CityGML specification.

## 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.*

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

<b>XML Schema Definition</b>	<b>Namespace identifier</b>	<b>Namespace prefix</b>
Geography Markup Language version 3.1.1 (from OGC)	<a href="http://www.opengis.net/gml">http://www.opengis.net/gml</a>	gml

<b>XML Schema Definition</b>	<b>Namespace identifier</b>	<b>Namespace prefix</b>
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	<a href="http://www.ascc.net/xml/schematron">http://www.ascc.net/xml/ schematron</a>	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).

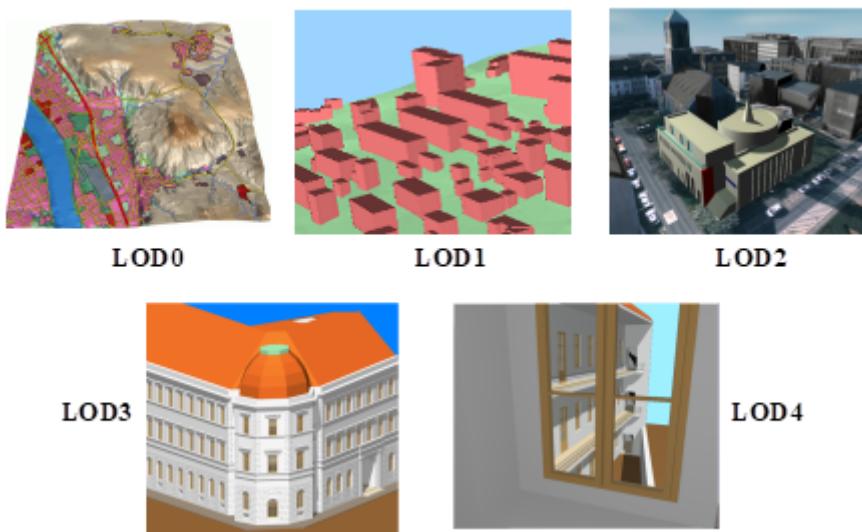


Figure 3. The five levels of detail (LOD) defined by CityGML (source: IGG Uni Bonn)

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  $\sigma$  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).

	<b>LOD0</b>	<b>LOD1</b>	<b>LOD2</b>	<b>LOD3</b>	<b>LOD4</b>
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

	<b>LOD0</b>	<b>LOD1</b>	<b>LOD2</b>	<b>LOD3</b>	<b>LOD4</b>
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.

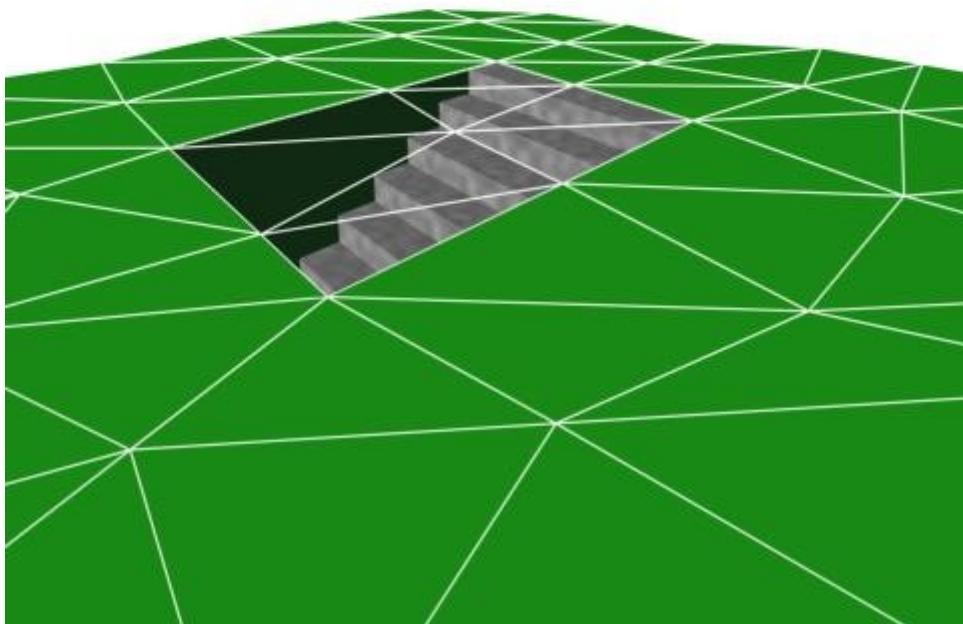
## 8.4. Closure surfaces

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.).

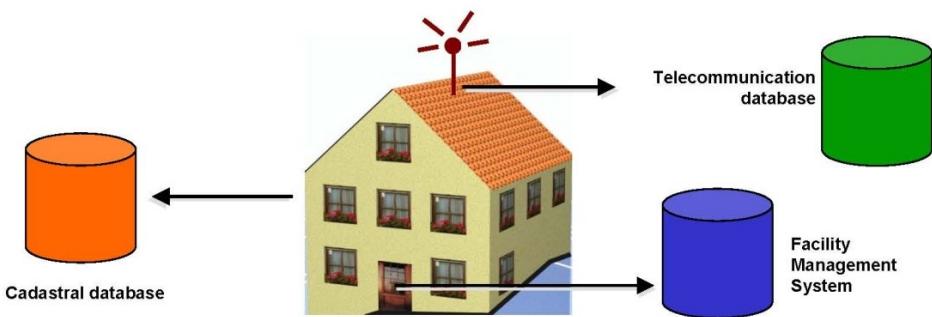


Figure 6. External references (graphic: IGG Uni Bonn).

## 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).

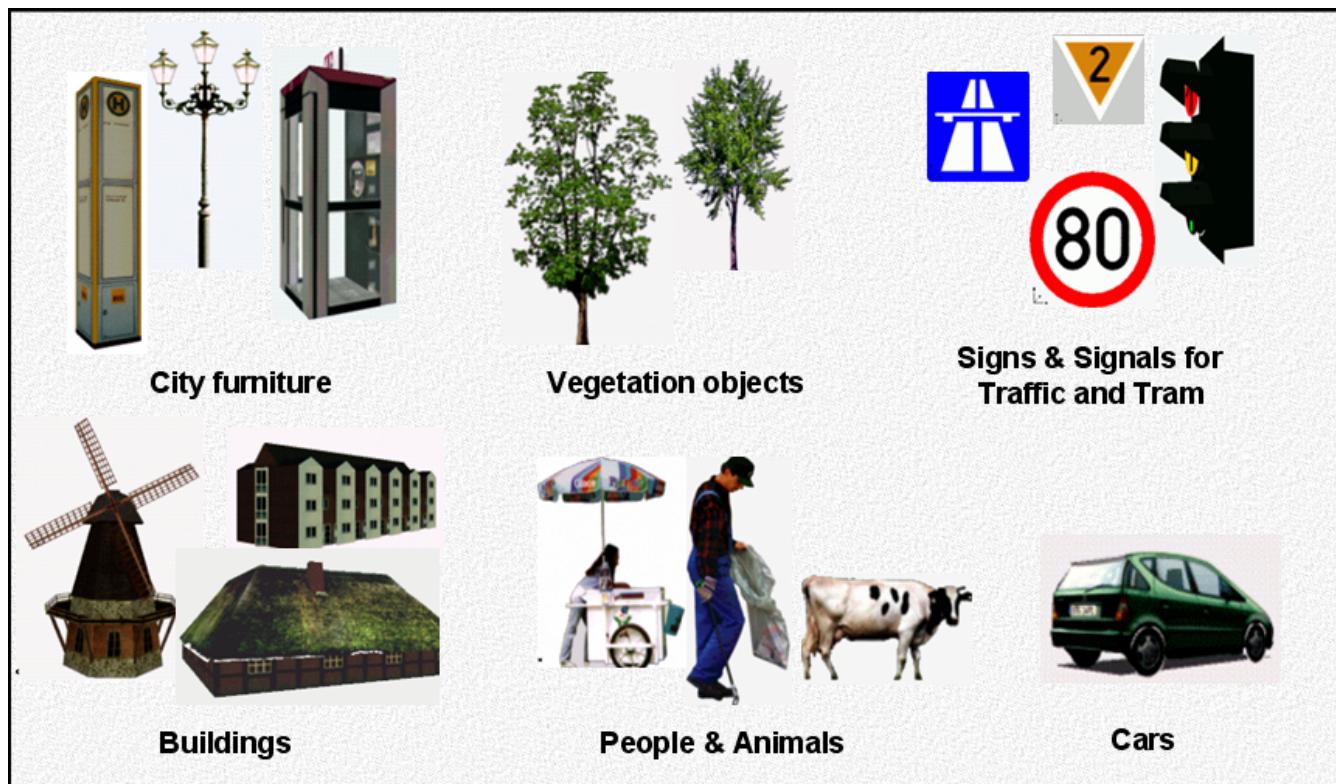
## 8.9. Appearances

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).



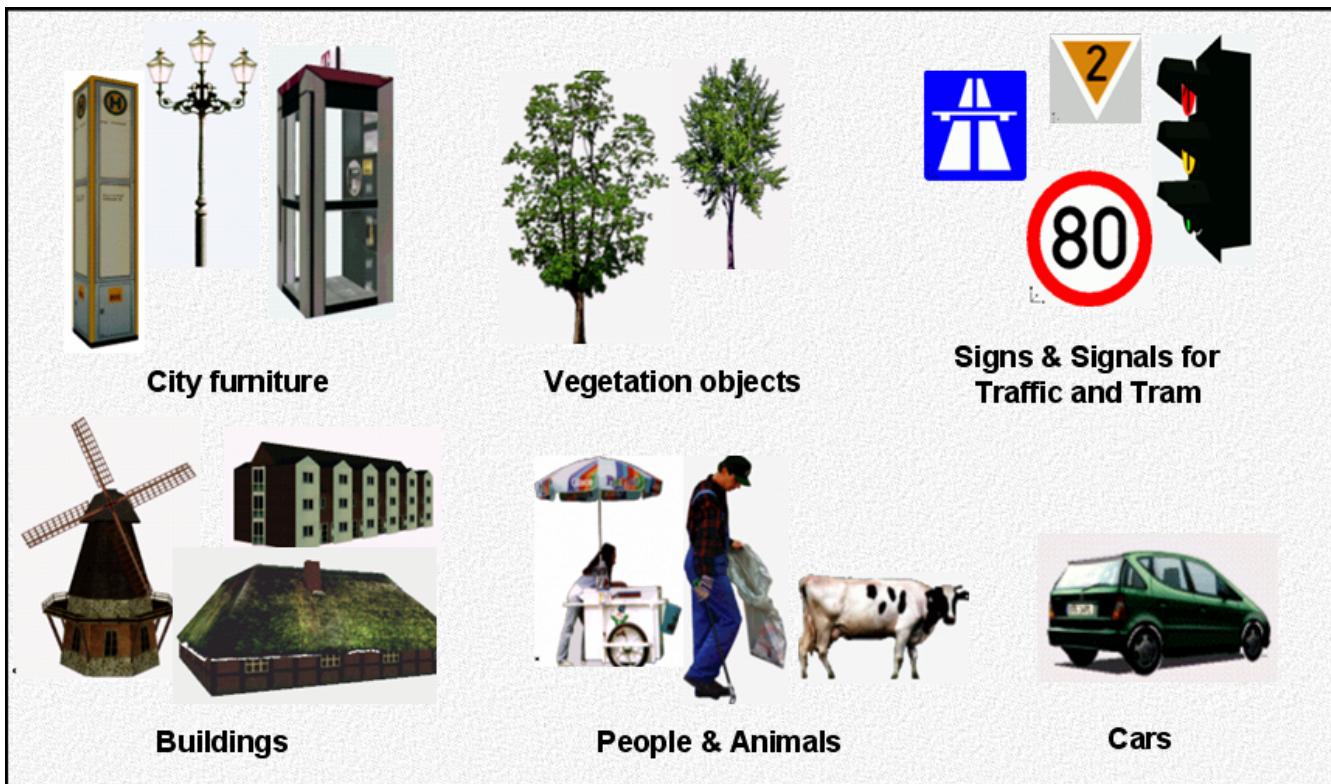


Figure 7. Examples of prototypic shapes (source: Rheinmetall Defence Electronics).

## 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

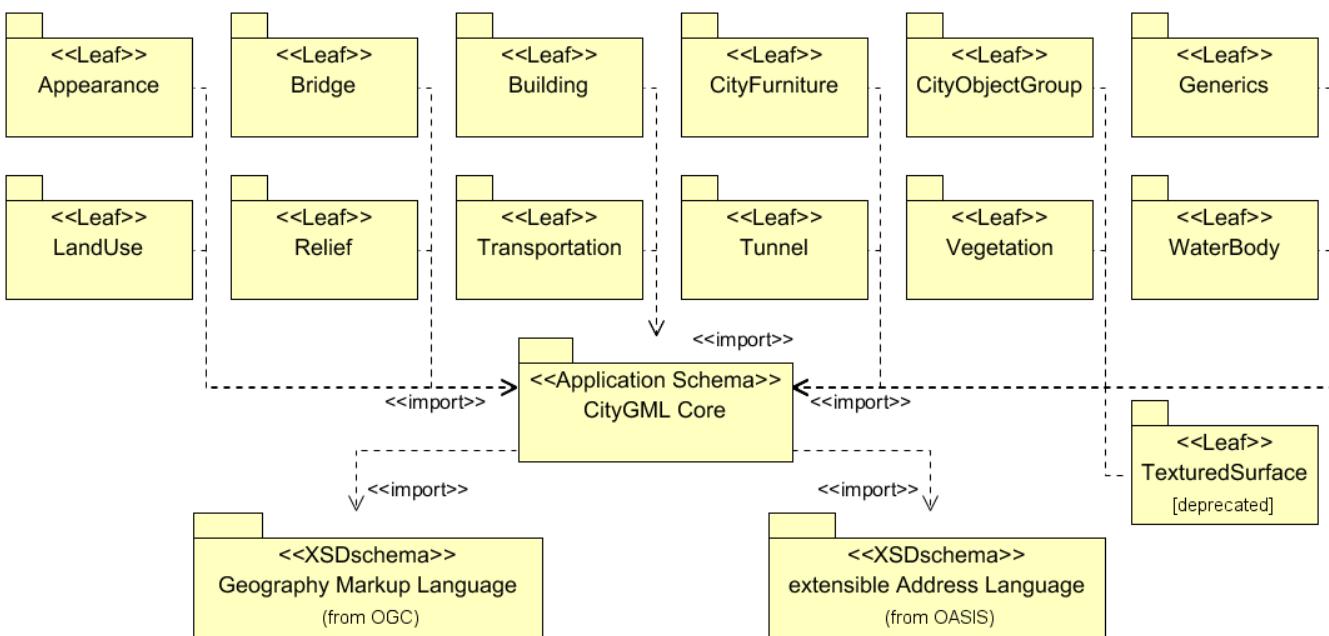
Each CityGML module is specified by its own XML Schema definition file and is defined within an individual and globally unique XML target namespace. According to dependency relations between modules, each module may, in addition, import namespaces associated to such related CityGML modules. However, 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	<a href="http://www.opengis.net/citygml/2.0">http://www.opengis.net/citygml/2.0</a>
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	<a href="http://www.opengis.net/citygml/appearance/2.0">http://www.opengis.net/citygml/appearance/2.0</a>
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	<a href="http://www.opengis.net/citygml/bridge/2.0">http://www.opengis.net/citygml/bridge/2.0</a>
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	<a href="http://www.opengis.net/citygml/building/2.0">http://www.opengis.net/citygml/building/2.0</a>
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	<a href="http://www.opengis.net/citygml/cityfurniture/2.0">http://www.opengis.net/citygml/cityfurniture/2.0</a>
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	<a href="http://www.opengis.net/citygml/cityobjectgroup/2.0">http://www.opengis.net/citygml/cityobjectgroup/2.0</a>
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	<a href="http://www.opengis.net/citygml/generics/2.0">http://www.opengis.net/citygml/generics/2.0</a>
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	<a href="http://www.opengis.net/citygml/landuse/2.0">http://www.opengis.net/citygml/landuse/2.0</a>
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	<a href="http://www.opengis.net/citygml/relief/2.0">http://www.opengis.net/citygml/relief/2.0</a>
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	<a href="http://www.opengis.net/citygml/transportation/2.0">http://www.opengis.net/citygml/transportation/2.0</a>
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	<a href="http://www.opengis.net/citygml/tunnel/2.0">http://www.opengis.net/citygml/tunnel/2.0</a>
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	<a href="http://www.opengis.net/citygml/vegetation/2.0">http://www.opengis.net/citygml/vegetation/2.0</a>
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	<a href="http://www.opengis.net/citygml/waterbody/2.0">http://www.opengis.net/citygml/waterbody/2.0</a>
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	<a href="http://www.opengis.net/citygml/texturesurface/2.0">http://www.opengis.net/citygml/texturesurface/2.0</a>
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](#).

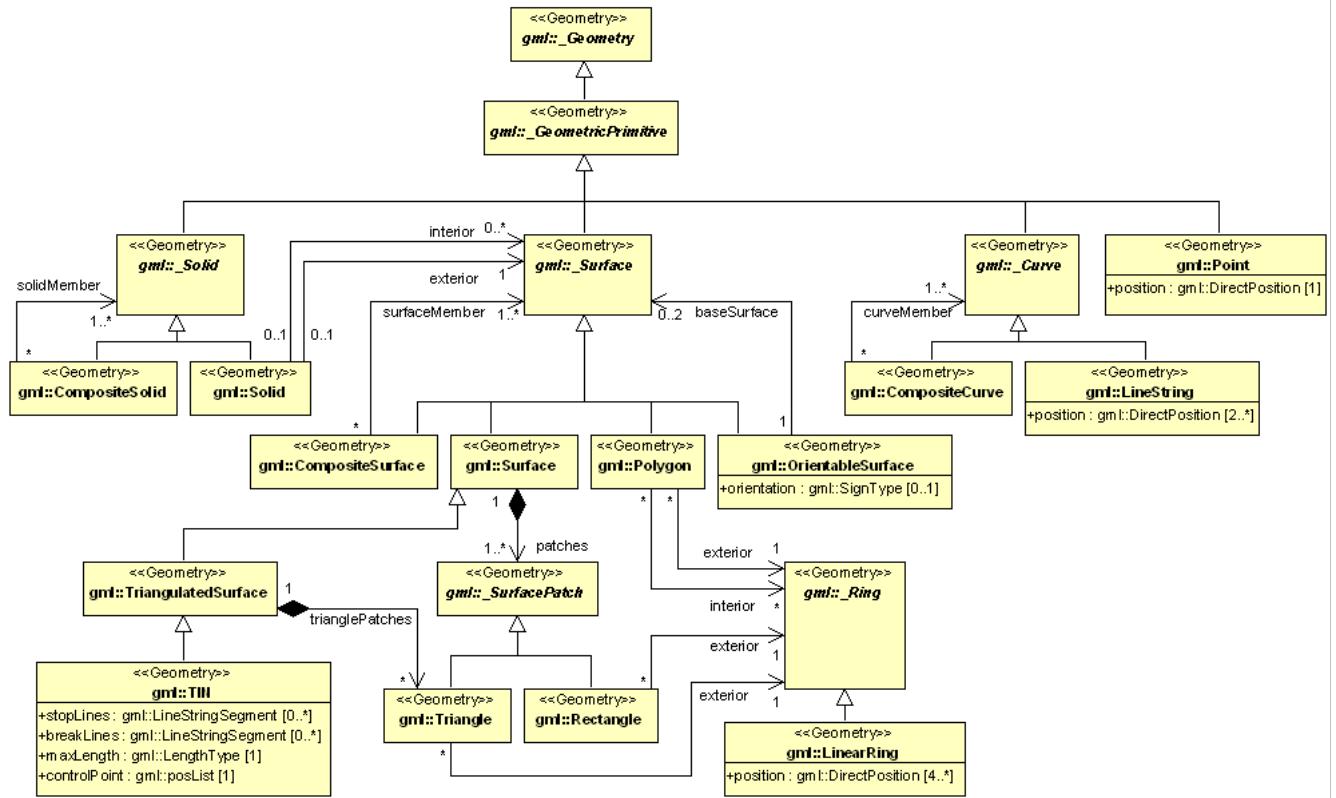


Figure 9. UML diagram of CityGML’s geometry model (subset and profile of GML3): Primitives and Composites.

### 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](#).

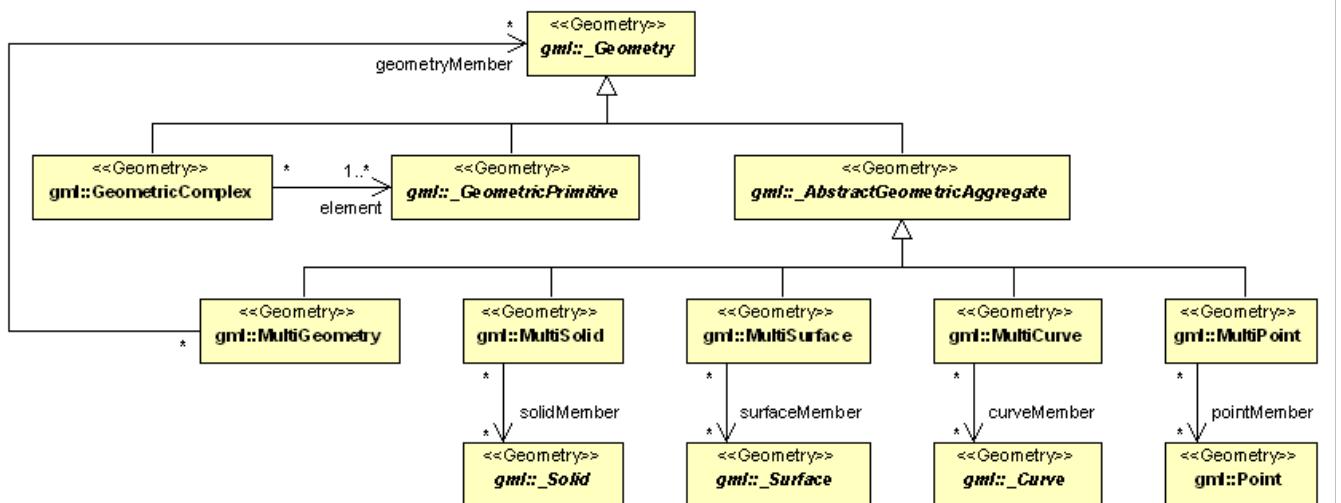


Figure 10. UML diagram of CityGML’s geometry model: Complexes and Aggregates

### 10.1.3. Combined Geometries

**NOTE** | short intro here

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

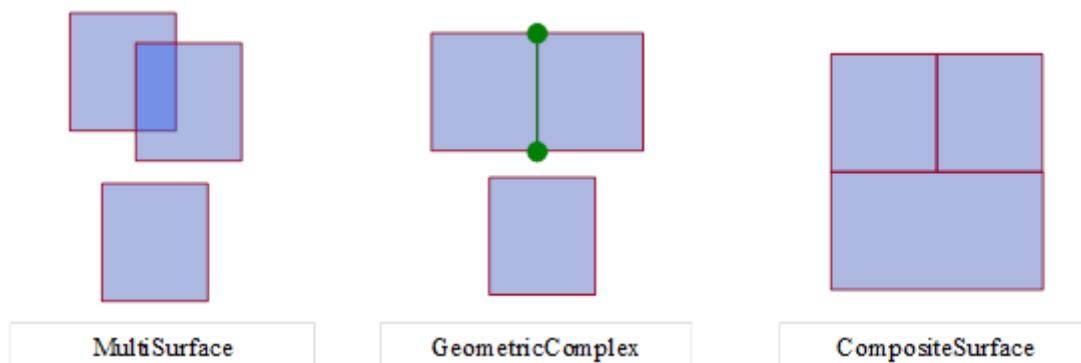


Figure 11. Combined geometries

#### 10.1.4. Recursive Aggregation

**NOTE** | short intro here

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

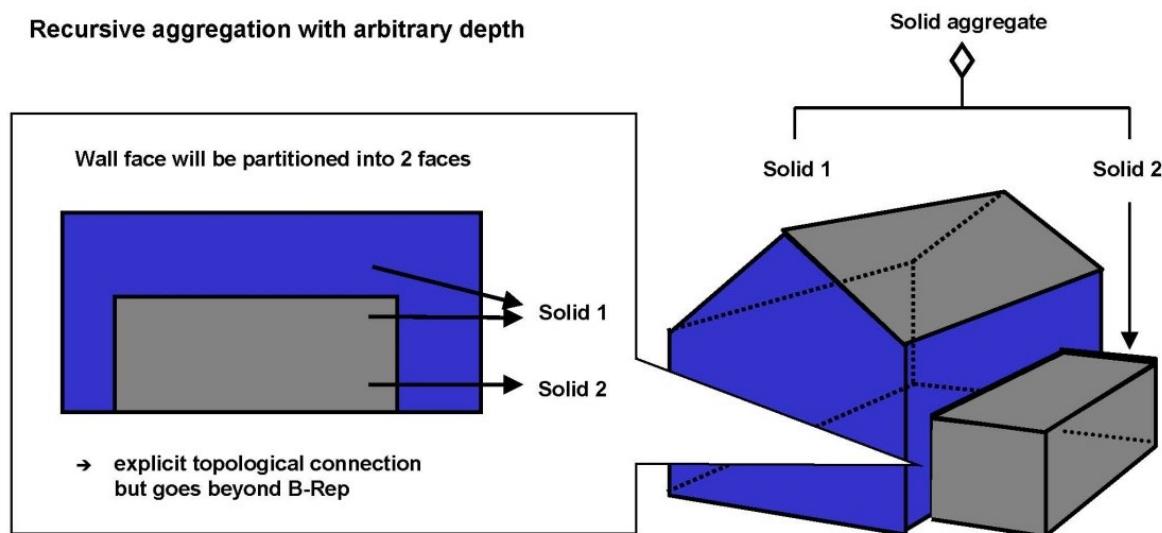


Figure 12. Recursive aggregation of objects and geometries in CityGML (graphic: IGG Uni Bonn).

## 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)

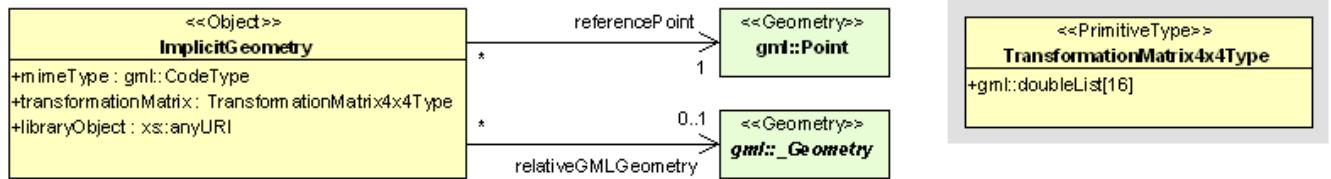


Figure 13. UML diagram of *ImplicitGeometries*. Prefixes are used to indicate XML namespaces associated with model elements. Element names without a prefix are defined within the CityGML Core module.

# Chapter 11. CityGML Conceptual Model

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

## 11.1. Appearance Model

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.1/req/req-class-appearance">http://www.opengis.net/spec/CityGML/3.1/req/req-class-appearance</a>	
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](#).

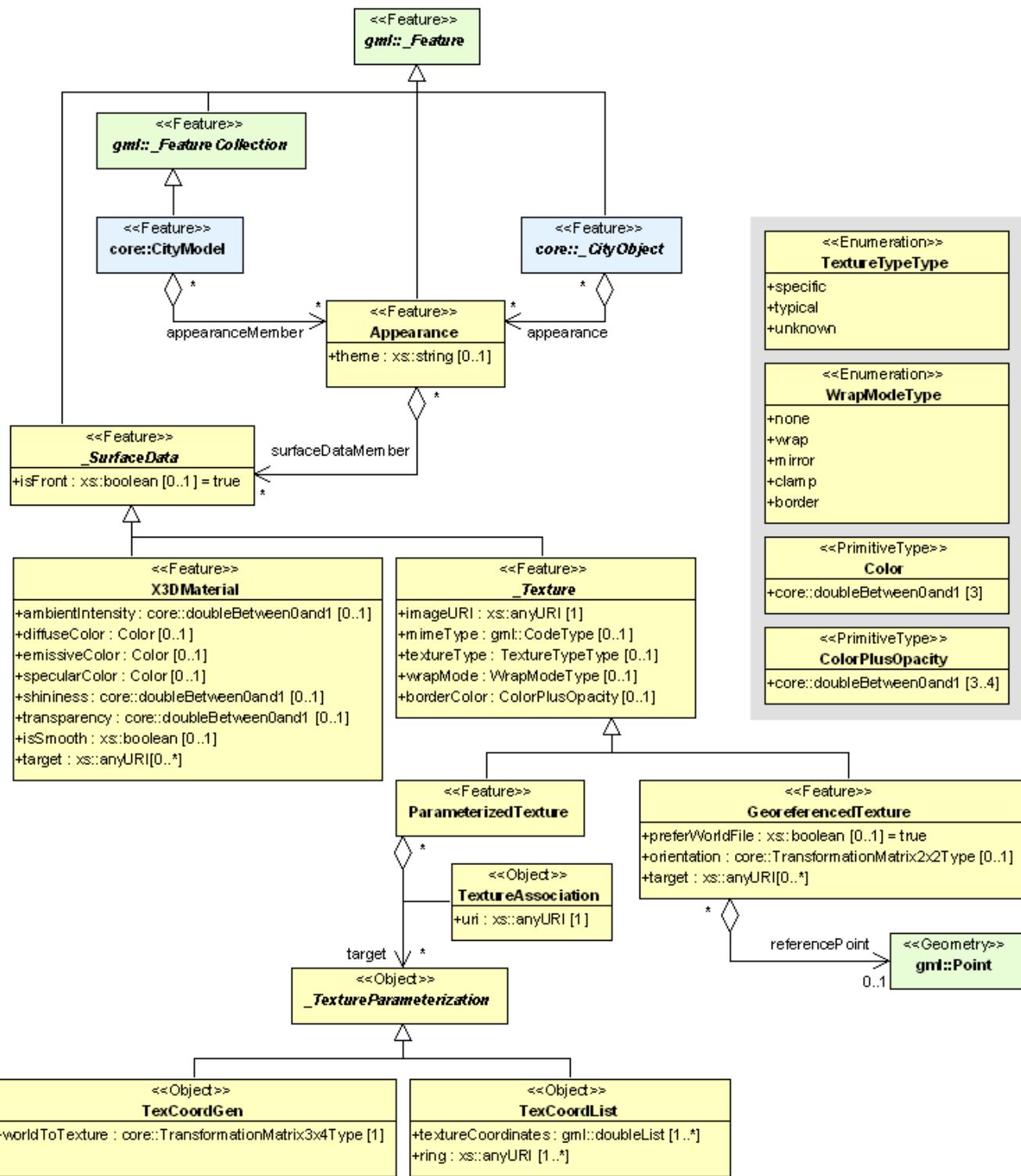


Figure 14. UML diagram of CityGML's appearance model.

### 11.1.1. Appearance Package

#### Package Appearance

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.1.2. Class AbstractSurfaceData

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

### Class AbstractSurfaceData

Definition:

Subclass Of: <--> section,>>

Stereotype: «FeatureType»

#### Roles

#### 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»

## 11.1.3. Class AbstractTexture

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

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 <a href="#">Appearance Data Dictionary</a> .
---	--

## Class AbstractTexture

Definition:

Subclass Of: [AbstractSurfaceData](#)

Stereotype: «FeatureType»

### Roles

### 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»

## 11.1.4. Class Appearance

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

### Class Appearance

Definition:

Subclass Of: [AbstractAppearance](#)

Stereotype: «FeatureType»

#### Roles

Role Name: surfaceData

Cardinality: \*

Target Class: [AbstractSurfaceData](#)

Definition: SIG3D: List of surface data properties for the current appearance theme

#### Attributes

Attribute Name: theme

Value Type: CharacterString

Definition: SIG3D: Theme name for all surfaceDataMembers. A theme is a catagory defining the semantics of the referenced surfaceDataMembers (e.g. infrared radiation,

Multiplicity: [0..1]

Stereotype: «Property»

### 11.1.5. Class Color

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 <a href="#">Appearance Data Dictionary</a> .

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

#### Class Color

Definition:

Subclass Of: [DoubleBetween0and1List](#)

Stereotype: «BasicType»

#### Roles

#### Attributes

### 11.1.6. Class ColorPlusOpacity

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

#### Class ColorPlusOpacity

Definition:
Subclass Of: <a href="#">DoubleBetween0and1List</a>
Stereotype: «BasicType»
<b>Roles</b>
<b>Attributes</b>

### 11.1.7. Class GeoreferencedTexture

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

Class GeoreferencedTexture
Definition:
Subclass Of: <a href="#">AbstractTexture</a>
Stereotype: «FeatureType»
<b>Roles</b>
Role Name: referencePoint
Cardinality: 0..1
Target Class: <a href="#">GM_Point</a>
Definition: SIG3D: Defines the location of the center of the upper left image pixel in world space.
<b>Attributes</b>

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»

### 11.1.8. Class ParameterizedTexture

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

#### Class ParameterizedTexture

Definition:  
 Subclass Of: [AbstractTexture](#)  
 Stereotype: «FeatureType»

<b>Roles</b>
Role Name: textureParameterization
Cardinality: *
Target Class: <a href="#">AbstractTextureParameterization</a>
Definition:
<b>Attributes</b>

### 11.1.9. Class TextureAssociation

<b>Requirement 8</b>	<a href="#">/req/Appearance/TextureAssociation</a>
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

<b>Class TextureAssociation</b>
Definition:
Subclass Of: <-> section,>>
Stereotype: «ObjectType»
<b>Roles</b>
<b>Attributes</b>
Attribute Name: uri
Value Type: URI
Definition: SIG3D: Link to the geometry object to be textured.
Multiplicity:
Stereotype: «Property»

## 11.1.10. Class X3DMaterial

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

Class X3DMaterial	
Definition:	
Subclass Of:	<a href="#">AbstractSurfaceData</a>
Stereotype:	«FeatureType»
Roles	
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»

### 11.1.11. Class AbstractTextureParameterization

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

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 <a href="#">Appearance Data Dictionary</a> .
---	--

### Class AbstractTextureParameterization

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

#### Attributes

### 11.1.12. Class TexCoordGen

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

### Class TexCoordGen

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

Role Name:	crs
Cardinality:	0..1
Target Class:	<b>SC_CRS</b>
Definition:	
<b>Attributes</b>	
<p>Attribute Name: worldToTexture</p> <p>Value Type: TransformationMatrix3x4</p> <p>Definition: SIG3D: 3x4 Matrix that defines the transformation between world coordinates and texture coordinates.</p> <p>Multiplicity:</p> <p>Stereotype: «Property»</p>	

### 11.1.13. Class TexCoordList

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

Class TexCoordList
Definition:
Subclass Of: <-- section,>>
Stereotype: «DataType»
<b>Roles</b>
<b>Attributes</b>

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»

### 11.1.14. Class TextureType

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

#### Class TextureType

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

#### Roles

#### 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:	

### 11.1.15. Class WrapMode

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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .
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 <a href="#">Appearance Data Dictionary</a> .

#### Class WrapMode

Definition:  
 Subclass Of: <-- section,>>  
 StereoType:

#### Roles

#### 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:	

### 11.1.16. 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](#).

## 11.2. Core

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.1/req/req-class-core">http://www.opengis.net/spec/CityGML/3.1/req/req-class-core</a>	
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](#).

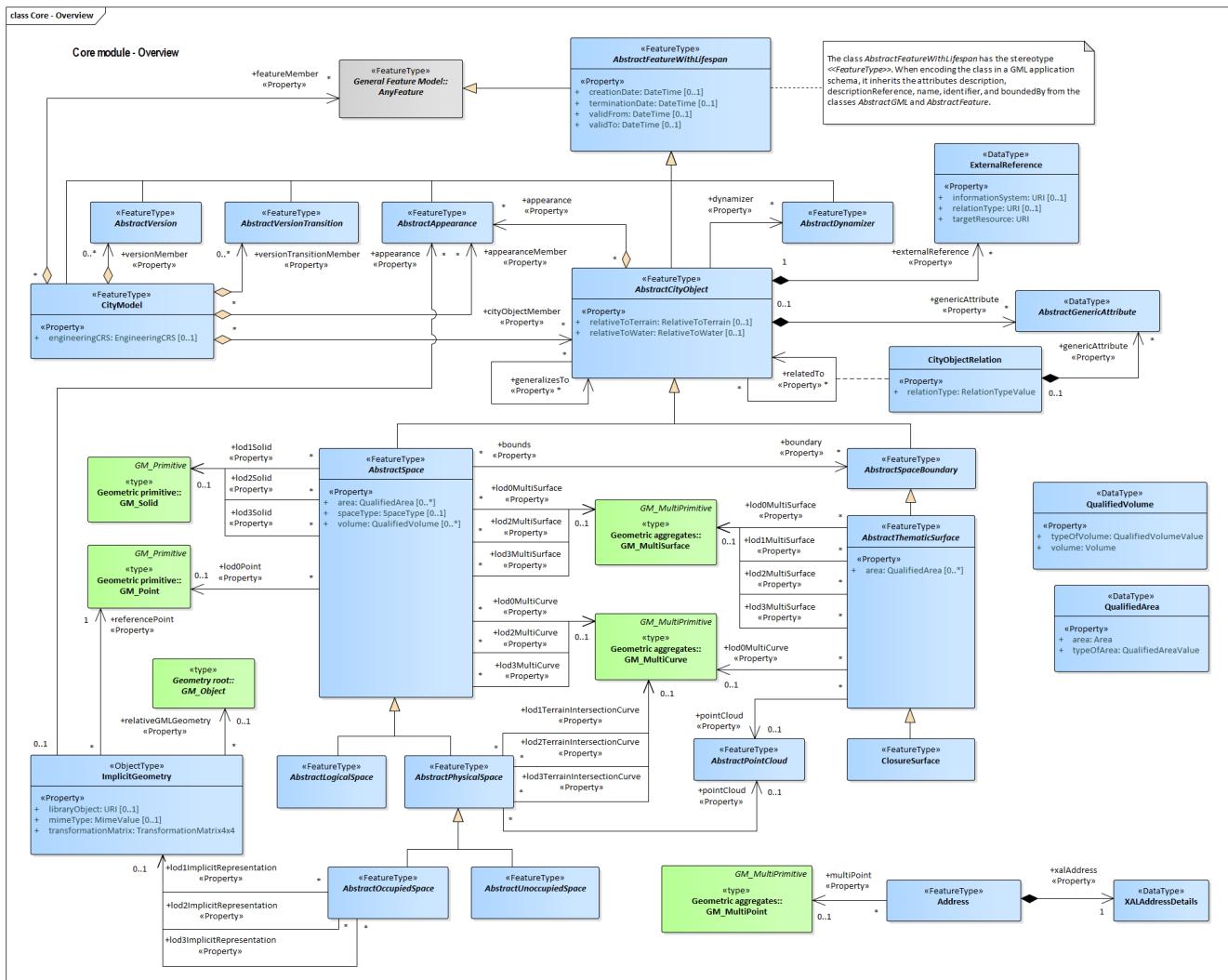


Figure 15. UML diagram of CityGML's core module.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.2.1. Core Package

### Package Core

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.2.2. Class AbstractAppearance

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class AbstractAppearance

Definition:

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

## 11.2.3. Class AbstractCityObject

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## Class AbstractCityObject

Definition:

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

### Roles

Role Name: appearance

Cardinality: \*

Target Class: [AbstractAppearance](#)

Definition:

Role Name: externalReference

Cardinality: \*

Target Class: [ExternalReference](#)

Definition:

Role Name: dynamizer

Cardinality: \*

Target Class: [AbstractDynamizer](#)

Definition:

Role Name: generalizesTo

Cardinality: \*

Target Class: [AbstractCityObject](#)

Definition:

Role Name: genericAttribute

Cardinality: \*

Target Class: [AbstractGenericAttribute](#)

Definition:

Role Name: relatedTo

Cardinality: \*

Target Class: [AbstractCityObject](#)

Definition:

### Attributes

Attribute Name: relativeToTerrain

Value Type: RelativeToTerrain

Definition: SIG3D: Vertical position of the AbstractCityObject relative to the surrounding terrain.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: relativeToWater

Value Type: RelativeToWater

Definition: SIG3D: Vertical position of the AbstractCityObject relative to a surrounding water surface.

Multiplicity: [0..1]

Stereotype: «Property»

#### 11.2.4. Class AbstractDynamizer

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

#### Class AbstractDynamizer

Definition:

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

Roles

Attributes

#### 11.2.5. Class AbstractFeatureWithLifespan

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## **Class AbstractFeatureWithLifespan**

Definition:

Subclass Of: [AnyFeature](#)

Stereotype: «FeatureType»

### **Roles**

### **Attributes**

Attribute Name: creationDate

Value Type: DateTime

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

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»

## **11.2.6. Class AbstractLogicalSpace**

<b>Requirement 19</b>	<b>/req/Core/AbstractLogicalSpace</b>
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

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 <a href="#">Core Data Dictionary</a> .
---	--

### Class AbstractLogicalSpace

Definition:

Subclass Of: [AbstractSpace](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.2.7. Class AbstractOccupiedSpace

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class AbstractOccupiedSpace

Definition:

Subclass Of: [AbstractPhysicalSpace](#)

Stereotype: «FeatureType»

#### Roles

Role Name: lod2ImplicitRepresentation

Cardinality: 0..1

Target Class: [ImplicitGeometry](#)

Definition: Relation to LOD2 implicit geometry of AbstractOccupiedSpace.

Role Name:	lod1ImplicitRepresentation
Cardinality:	0..1
Target Class:	<a href="#">ImplicitGeometry</a>
Definition:	Relation to LOD1 implicit geometry of AbstractOccupiedSpace.
Role Name:	lod3ImplicitRepresentation
Cardinality:	0..1
Target Class:	<a href="#">ImplicitGeometry</a>
Definition:	Relation to LOD3 implicit geometry of AbstractOccupiedSpace.
<b>Attributes</b>	

### 11.2.8. Class AbstractPhysicalSpace

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

<b>Class AbstractPhysicalSpace</b>
Definition:
Subclass Of: <a href="#">AbstractSpace</a>
Stereotype: «FeatureType»
<b>Roles</b>
Role Name: pointCloud
Cardinality: 0..1
Target Class: <a href="#">AbstractPointCloud</a>
Definition:
Role Name: lod3TerrainIntersectionCurve
Cardinality: 0..1
Target Class: <a href="#">GM_MultiCurve</a>
Definition: SIG 3D: Relation to intersection curve(s) between an LOD3 AbstractPhysicalSpace and a terrain (LOD1 or LOD2 or LOD3).

<p>Role Name: lod2TerrainIntersectionCurve</p> <p>Cardinality: 0..1</p> <p>Target Class: <a href="#">GM_MultiCurve</a></p> <p>Definition: SIG 3D: Relation to intersection curve(s) between an LOD2 AbstractPhysicalSpace and a terrain (LOD1 or LOD2 or LOD3).</p>
<p>Role Name: lod1TerrainIntersectionCurve</p> <p>Cardinality: 0..1</p> <p>Target Class: <a href="#">GM_MultiCurve</a></p> <p>Definition: SIG 3D: Relation to intersection curve(s) between an LOD1 AbstractPhysicalSpace and a terrain (LOD1 or LOD2 or LOD3).</p>
<b>Attributes</b>

### 11.2.9. Class AbstractPointCloud

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

#### Class AbstractPointCloud

Definition:  
Subclass Of: <-- section,>>  
StereoType: «FeatureType»

#### Roles

#### Attributes

### 11.2.10. Class AbstractSpace

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## Class AbstractSpace

Definition:

Subclass Of: [AbstractCityObject](#)

Stereotype: «FeatureType»

### Roles

Role Name: lod3MultiCurve

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relation to a LOD3 curve geometry of AbstractSpace.

Role Name: lod2MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relation to a LOD2 surface geometry of AbstractSpace.

Role Name: lod3Solid

Cardinality: 0..1

Target Class: [GM\\_Solid](#)

Definition: Relation to a LOD3 solid geometry of AbstractSpace.

Role Name: lod2Solid

Cardinality: 0..1

Target Class: [GM\\_Solid](#)

Definition: Relation to a LOD2 solid geometry of AbstractSpace.

Role Name: lod1Solid

Cardinality: 0..1

Target Class: [GM\\_Solid](#)

Definition: Relation to a LOD1 solid geometry of AbstractSpace.

Role Name:	lod0Point
Cardinality:	0..1
Target Class:	<a href="#">GM_Point</a>
Definition:	Relation to a LOD0 point geometry of AbstractSpace.
Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">AbstractSpaceBoundary</a>
Definition:	
Role Name:	lod0MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relation to a LOD0 surface geometry of AbstractSpace.
Role Name:	lod2MultiCurve
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiCurve</a>
Definition:	Relation to a LOD2 curve geometry of AbstractSpace.
Role Name:	lod3MultiSurface
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiSurface</a>
Definition:	Relation to a LOD3 surface geometry of AbstractSpace.
Role Name:	lod0MultiCurve
Cardinality:	0..1
Target Class:	<a href="#">GM_MultiCurve</a>
Definition:	Relation to a LOD0 curve geometry of AbstractSpace.
<b>Attributes</b>	
Attribute Name:	area
Value Type:	<a href="#">QualifiedArea</a>
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	spaceType
Value Type:	<a href="#">SpaceType</a>
Definition:	Degree of openness of an abstract space.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	volume
Value Type:	<a href="#">QualifiedVolume</a>
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

## 11.2.11. Class AbstractSpaceBoundary

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class AbstractSpaceBoundary

Definition:

Subclass Of: [AbstractCityObject](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

## 11.2.12. Class AbstractThematicSurface

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## Class AbstractThematicSurface

Definition:

Subclass Of: [AbstractSpaceBoundary](#)

Stereotype: «FeatureType»

### Roles

Role Name: lod1MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relation to a LOD1 surface geometry of AbstractThematicSurface.

Role Name: lod0MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relation to a LOD0 surface geometry of AbstractThematicSurface.

Role Name: lod2MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relation to a LOD2 surface geometry of AbstractThematicSurface.

Role Name: lod3MultiSurface

Cardinality: 0..1

Target Class: [GM\\_MultiSurface](#)

Definition: Relation to a LOD3 surface geometry of AbstractThematicSurface.

Role Name: lod0MultiCurve

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: Relation to a LOD0 curve geometry of AbstractThematicSurface.

Role Name: pointCloud

Cardinality: 0..1

Target Class: [AbstractPointCloud](#)

Definition:

### Attributes

Attribute Name: area

Value Type: QualifiedArea

Definition:

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.2.13. Class AbstractUnoccupiedSpace

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class AbstractUnoccupiedSpace

Definition:

Subclass Of: [AbstractPhysicalSpace](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.2.14. Class AbstractVersion

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class AbstractVersion

Definition:
Subclass Of: <a href="#">AbstractFeatureWithLifespan</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>

### 11.2.15. Class AbstractVersionTransition

<b>Requirement 28</b>	<a href="#">/req/Core/AbstractVersionTransition</a>
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

<b>Class AbstractVersionTransition</b>
Definition:
Subclass Of: <a href="#">AbstractFeatureWithLifespan</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>

### 11.2.16. Class Address

<b>Requirement 29</b>	<a href="#">/req/Core/Address</a>
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class Address

Definition:

Subclass Of: <-- section,>>

Stereotype: «FeatureType»

### Roles

Role Name: xalAddress

Cardinality: 1

Target Class: [XALAddressDetails](#)

Definition: SIG3D: Relation to an OASIS address object.

Role Name: multiPoint

Cardinality: 0..1

Target Class: [GM\\_MultiPoint](#)

Definition: SIG3D: Spatial reference of the address, e. g. positions of building entrances.

### Attributes

## 11.2.17. Class CityModel

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## Class CityModel

Definition:

Subclass Of: [AbstractFeatureWithLifespan](#)

Stereotype: «FeatureType»

### Roles

Role Name: cityObjectMember

Cardinality: \*

Target Class: [AbstractCityObject](#)

Definition: SIG3D: Relation of CityModel to AbstractTopLevelCityObject.

Role Name: versionMember

Cardinality: 0..\*

Target Class: [AbstractVersion](#)

Definition:

Role Name: versionTransitionMember

Cardinality: 0..\*

Target Class: [AbstractVersionTransition](#)

Definition:

Role Name: appearanceMember

Cardinality: \*

Target Class: [AbstractAppearance](#)

Definition:

Role Name: featureMember

Cardinality: \*

Target Class: [AnyFeature](#)

Definition:

### Attributes

Attribute Name: engineeringCRS

Value Type: EngineeringCRS

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

## 11.2.18. Class CityObjectRelation

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 <a href="#">Core Data Dictionary</a> .

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class CityObjectRelation

Definition:

Subclass Of: <-- section,>>

Stereotype:

#### Roles

Role Name: genericAttribute

Cardinality: \*

Target Class: [AbstractGenericAttribute](#)

Definition:

#### Attributes

Attribute Name: relationType

Value Type: RelationTypeValue

Definition:

Multiplicity:

Stereotype: «Property»

### 11.2.19. Class ClosureSurface

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class ClosureSurface

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

### Roles

### Attributes

## 11.2.20. Class DoubleBetween0and1

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class DoubleBetween0and1

Definition:

Subclass Of: <-> section,>>

Stereotype: «BasicType»

<b>Roles</b>
<b>Attributes</b>

### 11.2.21. Class DoubleBetween0and1List

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

#### Class DoubleBetween0and1List

Definition:

Subclass Of: <-- section,>>

Stereotype: «BasicType»

#### Roles

#### Attributes

Attribute Name: list

Value Type: DoubleBetween0and1

Definition:

Multiplicity:

Stereotype:

### 11.2.22. Class DoubleList

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class DoubleList

Definition:

Subclass Of: <-- section,>>

Stereotype: «BasicType»

### Roles

### Attributes

Attribute Name: list

Value Type: Real

Definition:

Multiplicity:

Stereotype:

## 11.2.23. Class ImplicitGeometry

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## Class ImplicitGeometry

Definition:

Subclass Of: <-- section,>>

Stereotype: «ObjectType»

### Roles

Role Name: appearance

Cardinality: \*

Target Class: [AbstractAppearance](#)

Definition:

Role Name: relativeGMLGeometry

Cardinality: 0..1

Target Class: [GM\\_Object](#)

Definition: SIG3D: Geometry of the prototype, specified in a local coordinate system.

Role Name: referencePoint

Cardinality: 1

Target Class: [GM\\_Point](#)

Definition: SIG3D: Base point of the object in the world coordinate system.

### Attributes

Attribute Name: libraryObject

Value Type: URI

Definition: SIG3D: External link to a prototype geometry.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: mimeType

Value Type: MimeValue

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»

## 11.2.24. Class IntegerBetween0and3

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class IntegerBetween0and3

Definition:  
 Subclass Of: <-- section,>>  
 StereoType: «BasicType»

#### Roles

#### Attributes

## 11.2.25. Class IntervalValue

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 <a href="#">Core Data Dictionary</a> .

## **Class IntervalValue**

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### **Roles**

### **Attributes**

## **11.2.26. Class MimeValue**

<b>Requirement 39</b>	<a href="#">/req/Core/MimeValue</a>
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 <a href="#">Core Data Dictionary</a> .

## **Class MimeValue**

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### **Roles**

### **Attributes**

## **11.2.27. Class OccupantTypeValue**

<b>Requirement 40</b>	<a href="#">/req/Core/OccupantTypeValue</a>
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 <a href="#">Core Data Dictionary</a> .

## **Class OccupantTypeValue**

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### **Roles**

### **Attributes**

## **11.2.28. Class OtherRelationTypeValue**

<b>Requirement 41</b>	<a href="#">/req/Core/OtherRelationTypeValue</a>
-----------------------	--

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 <a href="#">Core Data Dictionary</a> .
---	---

### Class OtherRelationTypeValue

Definition:

Subclass Of: [RelationTypeValue](#)

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.2.29. Class QualifiedAreaValue

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 <a href="#">Core Data Dictionary</a> .

### Class QualifiedAreaValue

Definition:

Subclass Of: <-- section,-->

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.2.30. Class QualifiedVolumeValue

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 <a href="#">Core Data Dictionary</a> .

### Class QualifiedVolumeValue

Definition:

Subclass Of: <-- section,-->

Stereotype: «CodeList»

#### Roles

## Attributes

### 11.2.31. Class RelationTypeValue

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 <a href="#">Core Data Dictionary</a> .

## Class RelationTypeValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

### 11.2.32. Class TemporalRelationTypeValue

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 <a href="#">Core Data Dictionary</a> .

## Class TemporalRelationTypeValue

Definition:

Subclass Of: [RelationTypeValue](#)

Stereotype: «CodeList»

### Roles

### Attributes

### 11.2.33. Class TopologicRelationTypeValue

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 <a href="#">Core Data Dictionary</a> .

## Class TopologicRelationTypeValue

Definition:

Subclass Of: [RelationTypeValue](#)

Stereotype: «CodeList»

Roles

Attributes

## 11.2.34. Class TransformationMatrix2x2

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

## Class TransformationMatrix2x2

Definition:

Subclass Of: [DoubleList](#)

Stereotype: «BasicType»

Roles

Attributes

## 11.2.35. Class TransformationMatrix3x4

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

#### Class TransformationMatrix3x4

Definition:

Subclass Of: [DoubleList](#)

Stereotype: «BasicType»

#### Roles

#### Attributes

### 11.2.36. Class TransformationMatrix4x4

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

D	The implementation Specification SHALL represent the attributes of all parent classes of the TransformationMatrix4x4 UML class; including the name, definition, type, and cardinality of those documented in the TransformationMatrix4x4 section of the <a href="#">Core Data Dictionary</a> .
---	--

#### Class TransformationMatrix4x4

Definition:

Subclass Of: [DoubleList](#)

Stereotype: «BasicType»

#### Roles

#### Attributes

### 11.2.37. Class AbstractGenericAttribute

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

#### Class AbstractGenericAttribute

Definition:

Subclass Of: <-> section,>>

Stereotype: «DataType»

#### Roles

#### Attributes

### 11.2.38. Class ExternalReference

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

#### Class ExternalReference

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

#### 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»

### 11.2.39. Class Occupancy

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the the Occupancy UML class; including the name, definition, type, and cardinality of those documented in the Occupancy section of the <a href="#">Core Data Dictionary</a> .

Class Occupancy
Definition:
Subclass Of: <-- section,>>
Stereotype: «DataType»
Roles
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»

## 11.2.40. Class QualifiedArea

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

### Class QualifiedArea

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

#### Attributes

Attribute Name: area

Value Type: Area

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: typeOfArea

Value Type: QualifiedAreaValue

Definition:

Multiplicity:

Stereotype: «Property»

## 11.2.41. Class QualifiedVolume

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

<b>Class QualifiedVolume</b>	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«DataType»
<b>Roles</b>	
<b>Attributes</b>	
Attribute Name:	typeOfVolume
Value Type:	QualifiedVolumeValue
Definition:	
Multiplicity:	
Stereotype:	«Property»
Attribute Name:	volume
Value Type:	Volume
Definition:	
Multiplicity:	
Stereotype:	«Property»

#### 11.2.42. Class RelativeToTerrain

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 <a href="#">Core Data Dictionary</a> .

## Class RelativeToTerrain

Definition:

Subclass Of: <-- section,>>

Stereotype:

### Roles

### 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:

## 11.2.43. Class RelativeToWater

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 RelativeToWater section of the <a href="#">Core Data Dictionary</a> .

## Class RelativeToWater

Definition:

Subclass Of: <-- section,>>

Stereotype:

## Roles

## 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 temporily is located above or below water level.

Multiplicity:

Stereotype:

## 11.2.44. Class SpaceType

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 <a href="#">Core Data Dictionary</a> .
---	--

## Class SpaceType

Definition:

Subclass Of: <-- section,>>

Stereotype:

### Roles

#### 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:

## 11.2.45. Class XALAddressDetails

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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .
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 <a href="#">Core Data Dictionary</a> .

D	<p>The implementation Specification SHALL represent the attributes of all parent classes of the XALAddressDetails UML class; including the name, definition, type, and cardinality of those documented in the XALAddressDetails section of the <a href="#">Core Data Dictionary</a>.</p>
---	--

### Class XALAddressDetails

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Roles

### Attributes

## 11.2.46. 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](#).

## 11.3. 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](#).

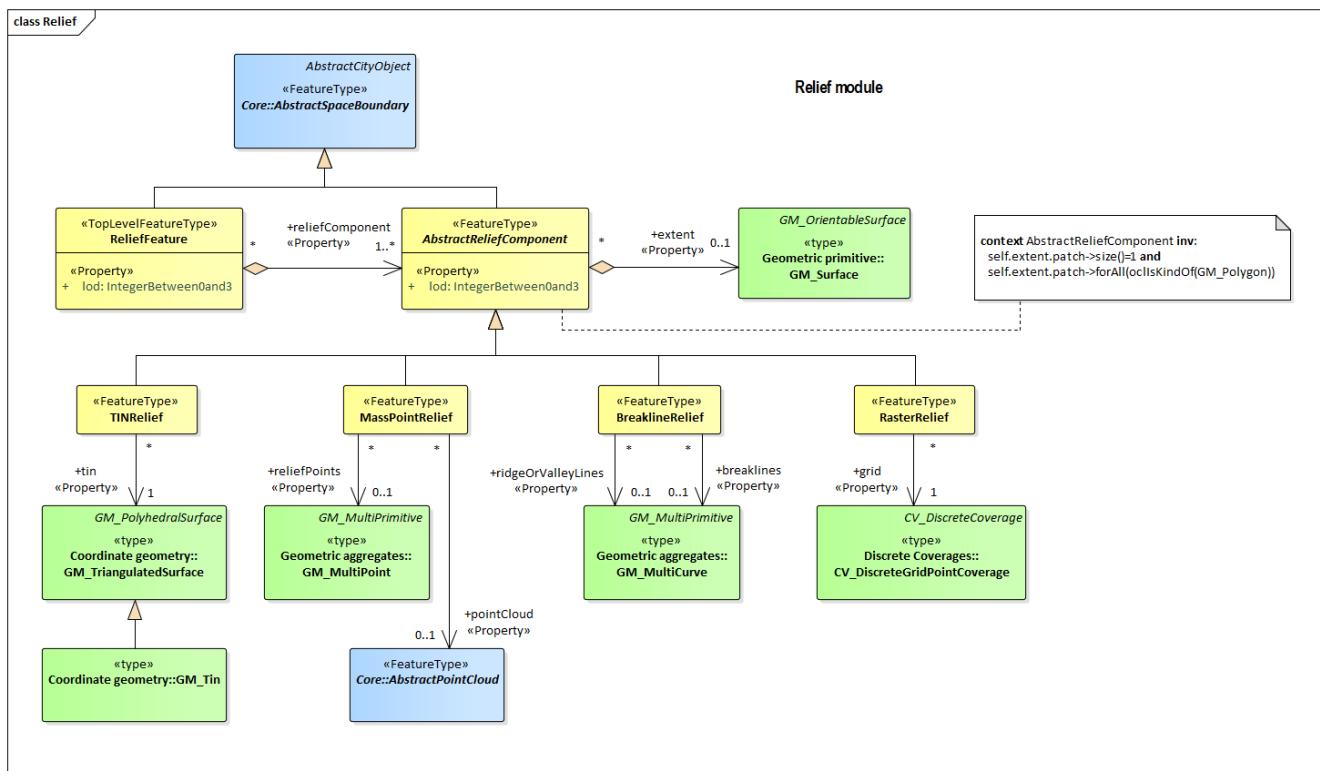


Figure 16. UML diagram of Relief module.

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

Yellow	indicates
Blue	indicates
Pink	indicates

### 11.3.1. Relief Package

#### Package Relief

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.3.2. Class AbstractReliefComponent

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 <a href="#">Relief Data Dictionary</a> .

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

### Class AbstractReliefComponent

Definition:

Subclass Of: [AbstractSpaceBoundary](#)

Stereotype: «FeatureType»

#### Roles

Role Name: extent

Cardinality: 0..1

Target Class: [GM\\_Surface](#)

Definition: SIG3D: Polygon (optionally with holes) denoting the extent where the relief component is valid.

#### 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»

### 11.3.3. Class BreaklineRelief

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

### Class BreaklineRelief

Definition:

Subclass Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

#### Roles

Role Name: ridgeOrValleyLines

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: SIG3D: Relation to 3D lines (gml:MultiCurve) denoting ridge or valley lines.

Role Name: breaklines

Cardinality: 0..1

Target Class: [GM\\_MultiCurve](#)

Definition: SIG3D: Relation to 3D lines (gml:MultiCurve) denoting break lines.

#### Attributes

### 11.3.4. Class MassPointRelief

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

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 <a href="#">Relief Data Dictionary</a> .
---	--

### Class MassPointRelief

Definition:

Subclass Of: [AbstractReliefComponent](#)

Stereotype: «FeatureType»

#### Roles

Role Name: reliefPoints

Cardinality: 0..1

Target Class: [GM\\_MultiPoint](#)

Definition:

Role Name: pointCloud

Cardinality: 0..1

Target Class: [AbstractPointCloud](#)

Definition:

#### Attributes

### 11.3.5. Class RasterRelief

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

### Class RasterRelief

Definition:
Subclass Of: <a href="#">AbstractReliefComponent</a>
Stereotype: «FeatureType»
<b>Roles</b>
Role Name: grid
Cardinality: 1
Target Class: <a href="#">CV_DiscreteGridPointCoverage</a>
Definition:
<b>Attributes</b>

### 11.3.6. Class ReliefFeature

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

<b>Class ReliefFeature</b>
Definition:
Subclass Of: <a href="#">AbstractSpaceBoundary</a>
Stereotype: «TopLevelFeatureType»
<b>Roles</b>
Role Name: reliefComponent
Cardinality: 1..*
Target Class: <a href="#">AbstractReliefComponent</a>
Definition: SIG3D: Relation of a ReliefFeature to its components.
<b>Attributes</b>

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»

### 11.3.7. Class TINRelief

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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .
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 <a href="#">Relief Data Dictionary</a> .

#### Class TINRelief

Definition:  
 Subclass Of: [AbstractReliefComponent](#)  
 Stereotype: «FeatureType»

#### Roles

Role Name: tin  
 Cardinality: 1  
 Target Class: [GM\\_TriangulatedSurface](#)  
 Definition:

#### Attributes

### 11.3.8. 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](#).

## 11.4. 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](#).

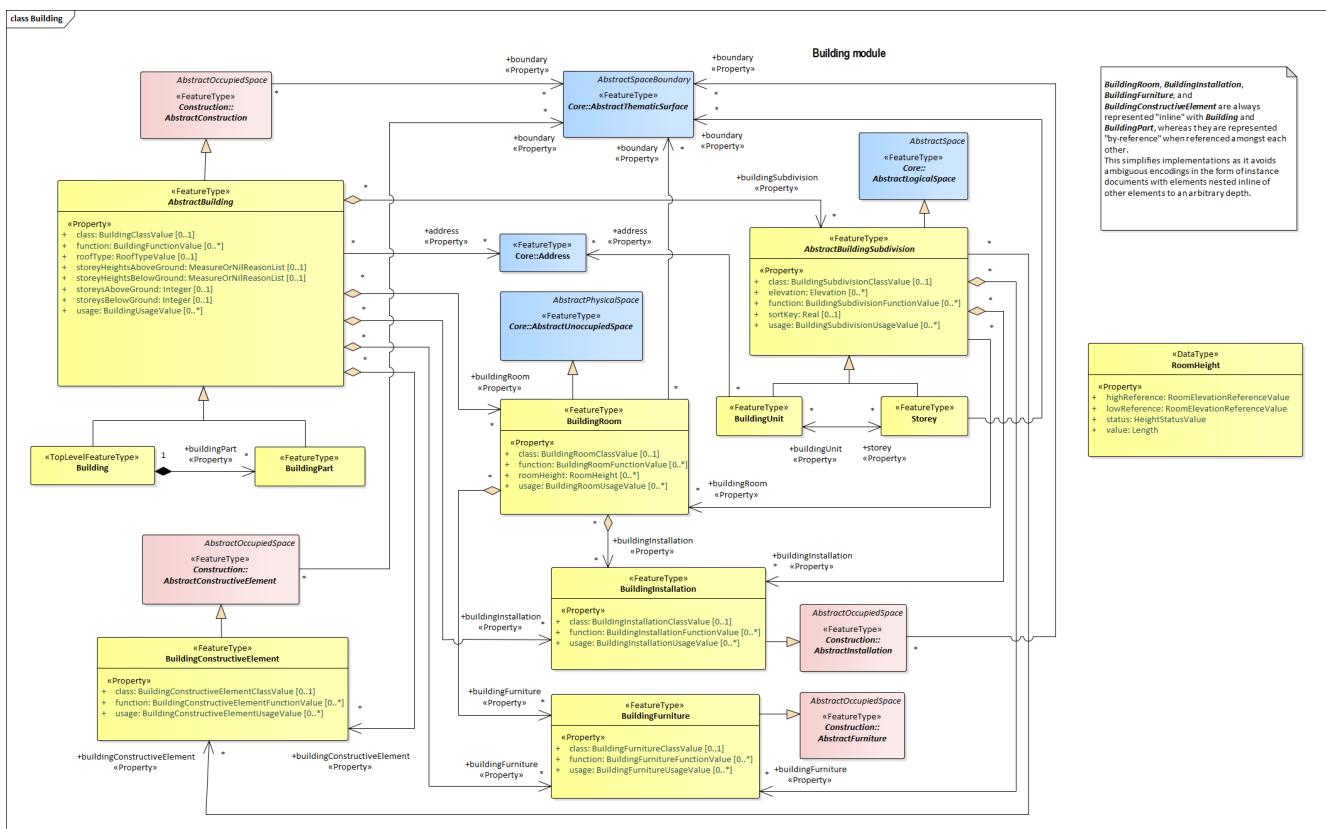


Figure 17. UML diagram of CityGML's building model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.4.1. Building Package

### Package Building

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.4.2. Class AbstractBuilding

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

### Class AbstractBuilding

Definition:

Subclass Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

### Roles

Role Name: buildingConstructiveElement

Cardinality: \*

Target Class: [BuildingConstructiveElement](#)

Definition:

Role Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: SIG3D: Relation between Building or BuildingPart and Address

Role Name: buildingSubdivision

Cardinality: \*

Target Class: [AbstractBuildingSubdivision](#)

Definition:

Role Name:	buildingFurniture
Cardinality:	*
Target Class:	<a href="#">BuildingFurniture</a>
Definition:	
Role Name:	buildingRoom
Cardinality:	*
Target Class:	<a href="#">BuildingRoom</a>
Definition:	SIG3D: Relation between Building or BuildingPart and Room.
Role Name:	buildingInstallation
Cardinality:	*
Target Class:	<a href="#">BuildingInstallation</a>
Definition:	SIG3D: Relation between Building or BuildingPart and BuildingInstallation.
<b>Attributes</b>	
Attribute Name:	class
Value Type:	<a href="#">BuildingClassValue</a>
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:	<a href="#">BuildingFunctionValue</a>
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:	<a href="#">RoofTypeValue</a>
Definition:	bSI: Basic configuration of the roof in terms of the different roof shapes.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeyHeightsAboveGround
Value Type:	<a href="#">MeasureOrNilReasonList</a>
Definition:	SIG3D: List of heights for each storey above ground.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	storeyHeightsBelowGround
Value Type:	<a href="#">MeasureOrNilReasonList</a>
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»

### 11.4.3. Class AbstractBuildingSubdivision

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

#### Class AbstractBuildingSubdivision

Definition:  
 Subclass Of: [AbstractLogicalSpace](#)  
 Stereotype: «FeatureType»

#### Roles

Role Name: buildingInstallation

Cardinality: \*

Target Class: [BuildingInstallation](#)

Definition:

Role Name: buildingFurniture

Cardinality: \*

Target Class: [BuildingFurniture](#)

Definition:

Role Name: buildingRoom

Cardinality: \*

Target Class: [BuildingRoom](#)

Definition:

Role Name: buildingConstructiveElement

Cardinality: \*

Target Class: [BuildingConstructiveElement](#)

Definition:

## 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»

## 11.4.4. Class Building

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

### Class Building

Definition:

Subclass Of: [AbstractBuilding](#)

Stereotype: «TopLevelFeatureType»

#### Roles

Role Name: buildingPart

Cardinality: \*

Target Class: [BuildingPart](#)

Definition: Relation between Building and BuildingPart.

#### Attributes

## 11.4.5. Class BuildingClassValue

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 <a href="#">Building Data Dictionary</a> .

### Class BuildingClassValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

## 11.4.6. Class BuildingConstructiveElement

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

<b>Class BuildingConstructiveElement</b>
Definition:
Subclass Of: <a href="#">AbstractConstructiveElement</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>
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»

#### 11.4.7. Class BuildingConstructiveElementClassValue

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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingConstructiveElementClassValue

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

#### Roles

#### Attributes

#### 11.4.8. Class BuildingConstructiveElementFunctionValue

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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingConstructiveElementFunctionValue

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

<b>Roles</b>
<b>Attributes</b>

### 11.4.9. Class BuildingConstructiveElementUsageValue

<b>Requirement 72</b>	/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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingConstructiveElementUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

### 11.4.10. Class BuildingFunctionValue

<b>Requirement 73</b>	/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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

### 11.4.11. Class BuildingFurniture

<b>Requirement 74</b>	/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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

## Class BuildingFurniture

Definition:

Subclass Of: [AbstractFurniture](#)

Stereotype: «FeatureType»

### Roles

### Attributes

Attribute Name: class

Value Type: BuildingFurnitureClassValue

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

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: BuildingFurnitureFunctionValue

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

Multiplicity: [0..\*]

Stereotype: «Property»

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»

#### 11.4.12. Class BuildingFurnitureClassValue

<b>Requirement 75</b>	/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 <a href="#">Building Data Dictionary</a> .

##### Class BuildingFurnitureClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.4.13. Class BuildingFurnitureFunctionValue

<b>Requirement 76</b>	/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 <a href="#">Building Data Dictionary</a> .

##### Class BuildingFurnitureFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.4.14. Class BuildingFurnitureUsageValue

<b>Requirement 77</b>	/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 <a href="#">Building Data Dictionary</a> .

##### Class BuildingFurnitureUsageValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.4.15. Class BuildingInstallation

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

<b>Class BuildingInstallation</b>
Definition:
Subclass Of: <a href="#">AbstractInstallation</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>
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..*]
Stereotype:	«Property»

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..*]
Stereotype:	«Property»

## 11.4.16. Class BuildingInstallationClassValue

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 <a href="#">Building Data Dictionary</a> .

### Class BuildingInstallationClassValue

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

#### Roles

#### Attributes

## 11.4.17. Class BuildingInstallationFunctionValue

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 <a href="#">Building Data Dictionary</a> .

### Class BuildingInstallationFunctionValue

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

<b>Roles</b>
<b>Attributes</b>

### 11.4.18. Class BuildingInstallationUsageValue

<b>Requirement 81</b>	/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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingInstallationUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.4.19. Class BuildingPart

<b>Requirement 82</b>	/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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingPart

Definition:

Subclass Of: [AbstractBuilding](#)

Stereotype: «FeatureType»

<b>Roles</b>	
<b>Attributes</b>	

### 11.4.20. Class BuildingRoom

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

<b>Class BuildingRoom</b>	
Definition:	
Subclass Of:	<a href="#">AbstractUnoccupiedSpace</a>
Stereotype:	«FeatureType»
<b>Roles</b>	
Role Name:	buildingFurniture
Cardinality:	*
Target Class:	<a href="#">BuildingFurniture</a>
Definition:	SIG3D: Relation between Room and BuildingFurniture
Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">AbstractThematicSurface</a>
Definition:	
Role Name:	buildingInstallation
Cardinality:	*
Target Class:	<a href="#">BuildingInstallation</a>
Definition:	SIG3D: Relation between Room and IntBuildingInstallation
<b>Attributes</b>	

Attribute Name:	class
Value Type:	BuildingRoomClassValue
Definition:	SIG3D: Classification of Room as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BuildingRoomFunctionValue
Definition:	SIG3D: Function of Room as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	roomHeight
Value Type:	RoomHeight
Definition:	Height of the room.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BuildingRoomUsageValue
Definition:	SIG3D: Usage of Room as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

## 11.4.21. Class BuildingRoomClassValue

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 <a href="#">Building Data Dictionary</a> .

Class BuildingRoomClassValue	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«CodeList»
Roles	
Attributes	

## 11.4.22. Class BuildingRoomFunctionValue

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 <a href="#">Building Data Dictionary</a> .
---	--

#### Class BuildingRoomFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.4.23. Class BuildingRoomUsageValue

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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingRoomUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.4.24. Class BuildingSubdivisionClassValue

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 <a href="#">Building Data Dictionary</a> .

#### Class BuildingSubdivisionClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

<b>Roles</b>
<b>Attributes</b>

### 11.4.25. Class BuildingSubdivisionFunctionValue

<b>Requirement 88</b>	/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 <a href="#">Building Data Dictionary</a> .

<b>Class BuildingSubdivisionFunctionValue</b>
Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.4.26. Class BuildingSubdivisionUsageValue

<b>Requirement 89</b>	/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 <a href="#">Building Data Dictionary</a> .

<b>Class BuildingSubdivisionUsageValue</b>
Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.4.27. Class BuildingUnit

<b>Requirement 90</b>	/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 <a href="#">Building Data Dictionary</a> .

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

### Class BuildingUnit

Definition:

Subclass Of: [AbstractBuildingSubdivision](#)

Stereotype: «FeatureType»

#### Roles

Role Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: SIG3D: Relation between BuildingUnit and Address.

#### Attributes

### 11.4.28. Class BuildingUsageValue

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 <a href="#">Building Data Dictionary</a> .

### Class BuildingUsageValue

Definition:

Subclass Of: <-> section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.4.29. Class RoofTypeValue

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 <a href="#">Building Data Dictionary</a> .
---	--

### Class RoofTypeValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.4.30. Class RoomElevationReferenceValue

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 <a href="#">Building Data Dictionary</a> .

### Class RoomElevationReferenceValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.4.31. Class Storey

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

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 <a href="#">Building Data Dictionary</a> .
---	--

### Class Storey

Definition:

Subclass Of: [AbstractBuildingSubdivision](#)

Stereotype: «FeatureType»

#### Roles

Role Name: buildingUnit

Cardinality: \*

Target Class: [BuildingUnit](#)

Definition:

Role Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

#### Attributes

### 11.4.32. Class RoomHeight

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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .
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 <a href="#">Building Data Dictionary</a> .

### Class RoomHeight

Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«DataType»
<b>Roles</b>	
<b>Attributes</b>	
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»

### 11.4.33. 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 (homo-geneous) 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*

<b>Geometric / semantic theme</b>	<b>Property type</b>	<b>LOD0</b>	<b>LOD1</b>	<b>LOD2</b>	<b>LOD3</b>	<b>LOD4</b>
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			•	•	•

<b>Geometric / semantic theme</b>	<b>Property type</b>	<b>LOD0</b>	<b>LOD1</b>	<b>LOD2</b>	<b>LOD3</b>	<b>LOD4</b>
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

### **11.4.34. 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.

## 11.4.35. 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) .

## 11.4.36. 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

### **11.4.37. 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 denote 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”.

## **11.5. Tunnel**

### **Requirements Class**

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

Target type	Conceptual Model
Dependency	TBD
Dependency	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](#).

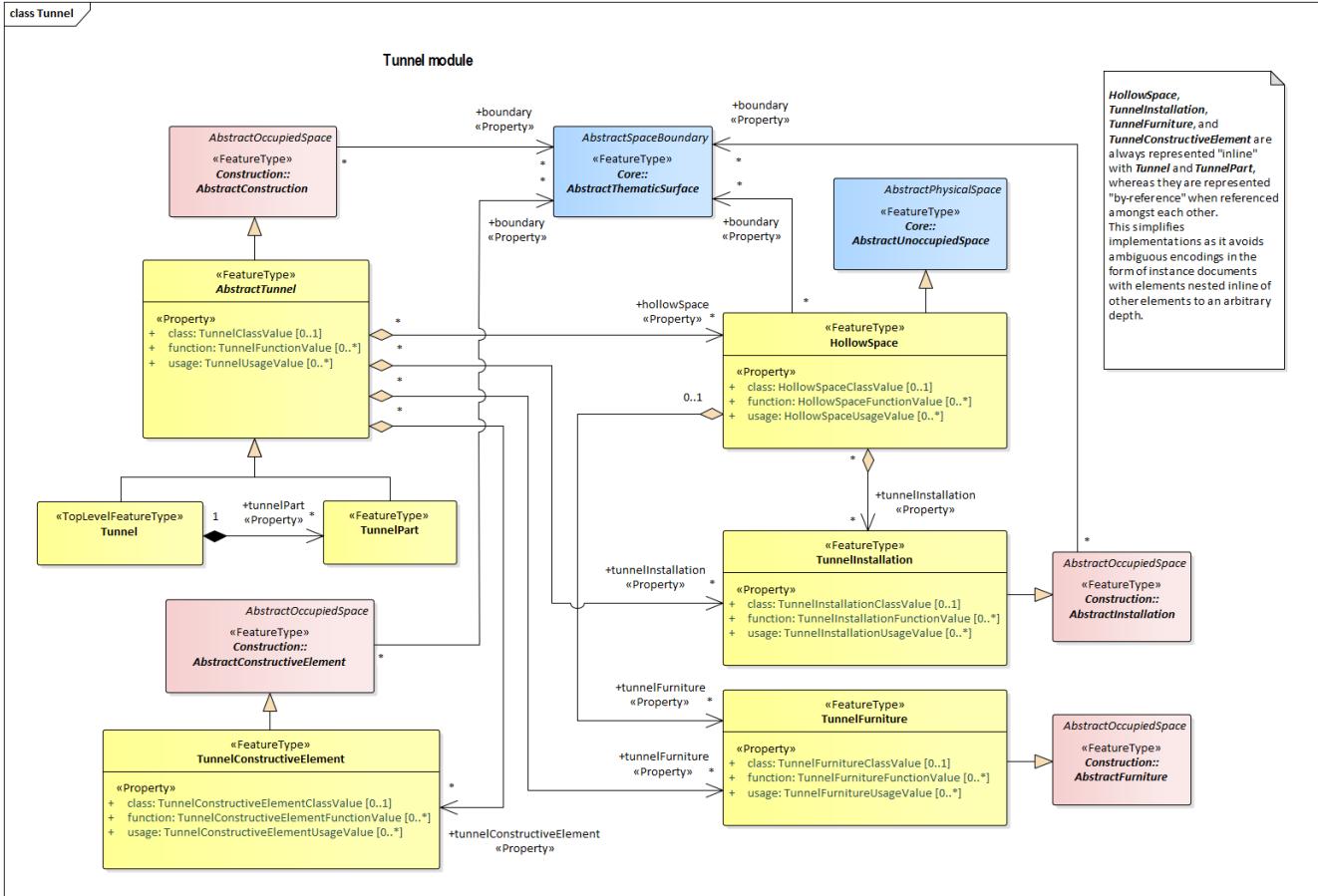


Figure 18. UML diagram of the Tunnel Model.

### 11.5.1. Tunnel Package

#### Package Tunnel

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.5.2. Class AbstractTunnel

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 <a href="#">Tunnel Data Dictionary</a> .

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

## Class AbstractTunnel

Definition:

Subclass Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

### Roles

Role Name: tunnelFurniture

Cardinality: \*

Target Class: [TunnelFurniture](#)

Definition:

Role Name: tunnelInstallation

Cardinality: \*

Target Class: [TunnelInstallation](#)

Definition: SIG3D: Relation between Tunnel or TunnelPart and TunnelInstallation.

Role Name: tunnelConstructiveElement

Cardinality: \*

Target Class: [TunnelConstructiveElement](#)

Definition:

Role Name: hollowSpace

Cardinality: \*

Target Class: [HollowSpace](#)

Definition: SIG3D: Relation between Tunnel or TunnelPart and HollowSpace.

### Attributes

Attribute Name: class

Value Type: TunnelClassValue

Definition: SIG3D: Classification of the actual usage of Tunnel or TunnelPart as given by the relevant national regulations, information communities, or specific applications.

Multiplicity: [0..1]

Stereotype: «Property»

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

### 11.5.3. Class HollowSpace

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

#### Class HollowSpace

Definition:  
 Subclass Of: [AbstractUnoccupiedSpace](#)  
 Stereotype: «FeatureType»

#### Roles

Role Name: boundary  
 Cardinality: \*  
 Target Class: [AbstractThematicSurface](#)  
 Definition:

Role Name:	tunnelFurniture
Cardinality:	*
Target Class:	<a href="#">TunnelFurniture</a>
Definition:	SIG3D: Relation between HollowSpace and TunnelFurniture.
Role Name:	tunnelInstallation
Cardinality:	*
Target Class:	<a href="#">TunnelInstallation</a>
Definition:	SIG3D: Relation between HollowSpace and TunnelInstallation.
<b>Attributes</b>	
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»

#### 11.5.4. Class HollowSpaceClassValue

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 <a href="#">Tunnel Data Dictionary</a> .

#### Class HollowSpaceClassValue

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

#### Roles

#### Attributes

### 11.5.5. Class HollowSpaceFunctionValue

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 <a href="#">Tunnel Data Dictionary</a> .

#### Class HollowSpaceFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

### 11.5.6. Class HollowSpaceUsageValue

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 <a href="#">Tunnel Data Dictionary</a> .

#### Class HollowSpaceUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

### 11.5.7. Class Tunnel

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

### Class Tunnel

Definition:

Subclass Of: [AbstractTunnel](#)

Stereotype: «TopLevelFeatureType»

#### Roles

Role Name: tunnelPart

Cardinality: \*

Target Class: [TunnelPart](#)

Definition: Relation between Tunnel and TunnelPart.

#### Attributes

### 11.5.8. Class TunnelClassValue

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 <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.5.9. Class TunnelConstructiveElement

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 <a href="#">Tunnel Data Dictionary</a> .

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelConstructiveElement

Definition:

Subclass Of: [AbstractConstructiveElement](#)

Stereotype: «FeatureType»

#### Roles

#### 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»

#### 11.5.10. Class TunnelConstructiveElementClassValue

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 <a href="#">Tunnel Data Dictionary</a> .
---	--

### Class TunnelConstructiveElementClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.5.11. Class TunnelConstructiveElementFunctionValue

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 <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelConstructiveElementFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.5.12. Class TunnelConstructiveElementUsageValue

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 <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelConstructiveElementUsageValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.5.13. Class TunnelFunctionValue

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 <a href="#">Tunnel Data Dictionary</a> .

<b>Class TunnelFunctionValue</b>
Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.5.14. Class TunnelFurniture

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

## Class TunnelFurniture

Definition:

Subclass Of: [AbstractFurniture](#)

Stereotype: «FeatureType»

### Roles

### 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»

## 11.5.15. Class TunnelFurnitureClassValue

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 <a href="#">Tunnel Data Dictionary</a> .

## Class TunnelFurnitureClassValue

Definition:

Subclass Of: <-> section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.5.16. Class TunnelFurnitureFunctionValue

<b>Requirement 110</b>	/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 <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelFurnitureFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.5.17. Class TunnelFurnitureUsageValue

<b>Requirement 111</b>	/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 <a href="#">Tunnel Data Dictionary</a> .

### Class TunnelFurnitureUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.5.18. Class TunnelInstallation

<b>Requirement 112</b>	/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 <a href="#">Tunnel Data Dictionary</a> .

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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

## Class TunnelInstallation

Definition:

Subclass Of: [AbstractInstallation](#)

Stereotype: «FeatureType»

### Roles

### 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»

## 11.5.19. Class TunnelInstallationClassValue

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 <a href="#">Tunnel Data Dictionary</a> .
---	---

#### Class TunnelInstallationClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.5.20. Class TunnelInstallationFunctionValue

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 <a href="#">Tunnel Data Dictionary</a> .

#### Class TunnelInstallationFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.5.21. Class TunnelInstallationUsageValue

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 <a href="#">Tunnel Data Dictionary</a> .

#### Class TunnelInstallationUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

<b>Roles</b>
<b>Attributes</b>

### 11.5.22. Class TunnelPart

<b>Requirement 116</b>	/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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .
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 <a href="#">Tunnel Data Dictionary</a> .

<b>Class TunnelPart</b>
Definition:
Subclass Of: <a href="#">AbstractTunnel</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>

### 11.5.23. Class TunnelUsageValue

<b>Requirement 117</b>	/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 <a href="#">Tunnel Data Dictionary</a> .

<b>Class TunnelUsageValue</b>
Definition:
Subclass Of: <-> section,>>
Stereotype: «CodeList»

<b>Roles</b>
<b>Attributes</b>

### 11.5.24. 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](#).

## 11.6. Bridge Model

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.0/req/req-class-bridge">http://www.opengis.net/spec/CityGML/3.0/req/req-class-bridge</a>	
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](#).

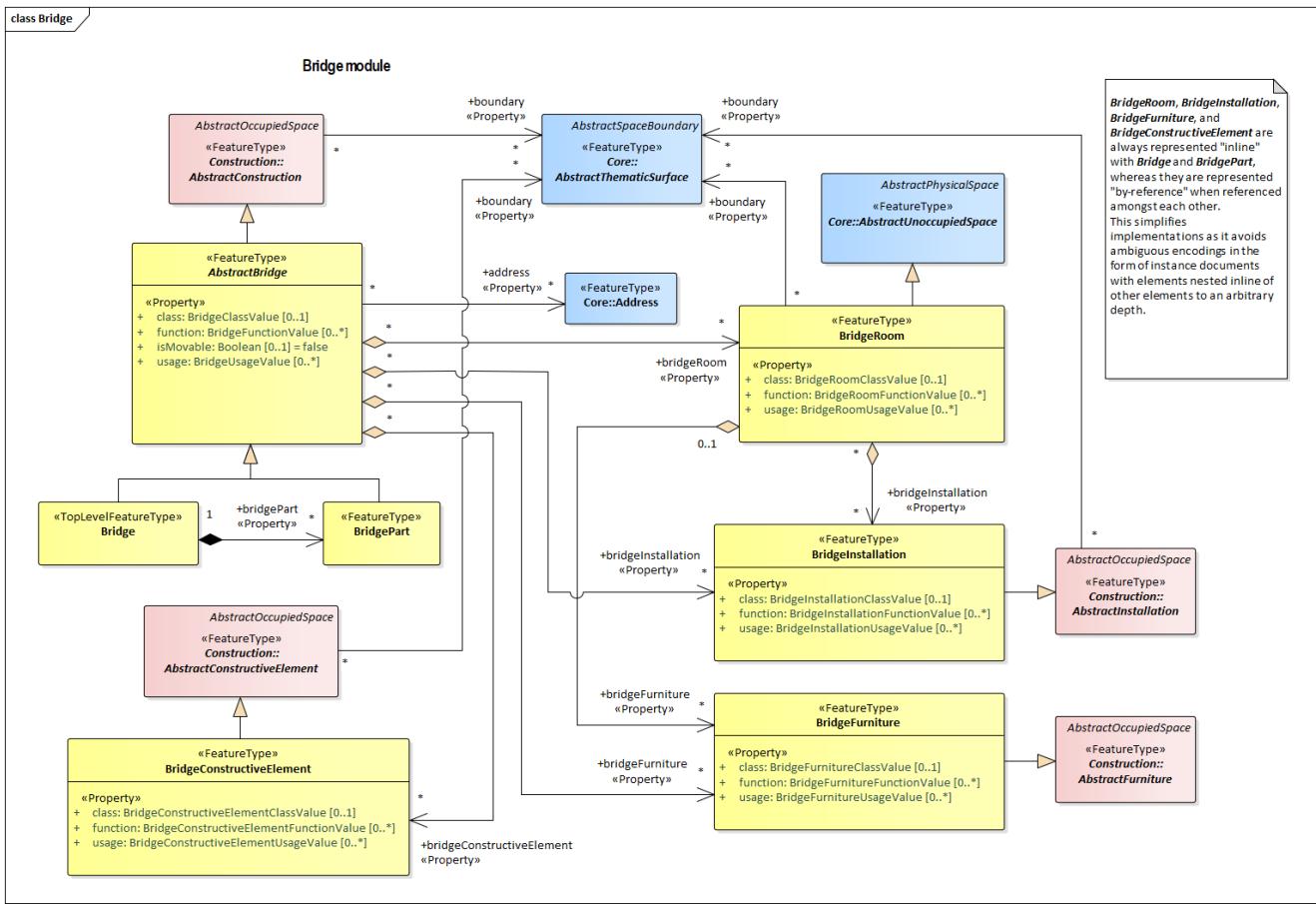


Figure 19. UML diagram of the Bridge Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.6.1. Bridge Package

### Package Bridge

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.6.2. Class AbstractBridge

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 <a href="#">Bridge Data Dictionary</a> .

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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

## Class AbstractBridge

Definition:

Subclass Of: [AbstractConstruction](#)

Stereotype: «FeatureType»

### Roles

Role Name: bridgeRoom

Cardinality: \*

Target Class: [BridgeRoom](#)

Definition: SIG3D: Relation between Bridge or BridgePart and BridgeRoom

Role Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: SIG3D: Relation between Bridge or BridgePart and Address

Role Name: bridgeConstructiveElement

Cardinality: \*

Target Class: [BridgeConstructiveElement](#)

Definition: SIG3D: Relation between Bridge or BridgePart and BridgeConstructionElement

Role Name: bridgeInstallation

Cardinality: \*

Target Class: [BridgeInstallation](#)

Definition: SIG3D: Relation between Bridge or BridgePart and BridgeInstallation

Role Name: bridgeFurniture

Cardinality: \*

Target Class: [BridgeFurniture](#)

Definition:

### Attributes

Attribute Name: class

Value Type: [BridgeClassValue](#)

Definition: SIG3D: Classification of the actual usage of Bridge or BridgePart as given by the relevant national regulations, information communities, or specific applications.

Multiplicity: [0..1]

Stereotype: «Property»

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

### 11.6.3. Class Bridge

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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

Class Bridge
Definition:
Subclass Of: <a href="#">AbstractBridge</a>
Stereotype: «TopLevelFeatureType»
Roles

Role Name:	bridgePart
Cardinality:	*
Target Class:	<a href="#">BridgePart</a>
Definition:	Relation between Bridge and BridgePart.

#### Attributes

### 11.6.4. Class BridgeClassValue

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 <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.6.5. Class BridgeConstructiveElement

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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

## Class BridgeConstructiveElement

Definition:

Subclass Of: [AbstractConstructiveElement](#)

Stereotype: «FeatureType»

### Roles

### 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»

## 11.6.6. Class BridgeConstructiveElementClassValue

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 <a href="#">Bridge Data Dictionary</a> .

## Class BridgeConstructiveElementClassValue

Definition:

Subclass Of: <-> section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.6.7. Class BridgeConstructiveElementFunctionValue

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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeConstructiveElementFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

## 11.6.8. Class BridgeConstructiveElementUsageValue

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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeConstructiveElementUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

## 11.6.9. Class BridgeFunctionValue

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 <a href="#">Bridge Data Dictionary</a> .

## Class BridgeFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.6.10. Class BridgeFurniture

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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

## Class BridgeFurniture

Definition:

Subclass Of: [AbstractFurniture](#)

Stereotype: «FeatureType»

### Roles

### 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»

## 11.6.11. Class BridgeFurnitureClassValue

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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeFurnitureClassValue

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

#### Roles

#### Attributes

## 11.6.12. Class BridgeFurnitureFunctionValue

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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeFurnitureFunctionValue

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

<b>Roles</b>
<b>Attributes</b>

### 11.6.13. Class BridgeFurnitureUsageValue

<b>Requirement 129</b>	/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 <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeFurnitureUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.6.14. Class BridgeInstallation

<b>Requirement 130</b>	/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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

#### Class BridgeInstallation

Definition:	
Subclass Of:	<a href="#">AbstractInstallation</a>
Stereotype:	«FeatureType»
<b>Roles</b>	
<b>Attributes</b>	
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»

## 11.6.15. Class BridgeInstallationClassValue

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 <a href="#">Bridge Data Dictionary</a> .

<b>Class BridgeInstallationClassValue</b>	
Definition:	
Subclass Of:	<-- section,>
Stereotype:	«CodeList»
<b>Roles</b>	
<b>Attributes</b>	

## 11.6.16. Class BridgeInstallationFunctionValue

<b>Requirement 132</b>	/req/Bridge/BridgeInstallationFunctionValue
A	Any use of the BridgeInstallationFunctionValue type in an Implementation Specification SHALL be restricted to the valid values specified in the BridgeInstallationFunctionValue UML class as documented in the BridgeInstallationFunctionValue section of the <a href="#">Bridge Data Dictionary</a> .

### Class BridgeInstallationFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

## 11.6.17. Class BridgeInstallationUsageValue

<b>Requirement 133</b>	/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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeInstallationUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

## 11.6.18. Class BridgePart

<b>Requirement 134</b>	/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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgePart

Definition:

Subclass Of: [AbstractBridge](#)

Stereotype: «FeatureType»

### Roles

### Attributes

## 11.6.19. Class BridgeRoom

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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .
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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeRoom

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

Role Name:	bridgeInstallation
Cardinality:	*
Target Class:	<a href="#">BridgeInstallation</a>
Definition:	SIG3D: Relation between BridgeRoom and BridgegInstallation
Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">AbstractThematicSurface</a>
Definition:	
Role Name:	bridgeFurniture
Cardinality:	*
Target Class:	<a href="#">BridgeFurniture</a>
Definition:	SIG3D: Relation between BridgeRoom and BridgeFurniture
<b>Attributes</b>	
Attribute Name:	class
Value Type:	BridgeRoomClassValue
Definition:	SIG3D: Classification of BridgeRoom as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
Attribute Name:	function
Value Type:	BridgeRoomFunctionValue
Definition:	SIG3D: Specified function of BridgeRoom as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	usage
Value Type:	BridgeRoomUsageValue
Definition:	SIG3D: Actual usage of BridgeRoom as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..*]
Stereotype:	«Property»

## 11.6.20. Class BridgeRoomClassValue

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 <a href="#">Bridge Data Dictionary</a> .

### Class BridgeRoomClassValue

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

**Roles**

**Attributes**

### 11.6.21. Class BridgeRoomFunctionValue

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 <a href="#">Bridge Data Dictionary</a> .

**Class BridgeRoomFunctionValue**

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

**Roles**

**Attributes**

### 11.6.22. Class BridgeRoomUsageValue

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 <a href="#">Bridge Data Dictionary</a> .

**Class BridgeRoomUsageValue**

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

**Roles**

**Attributes**

### 11.6.23. Class BridgeUsageValue

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 <a href="#">Bridge Data Dictionary</a> .
---	---

## Class BridgeUsageValue

Definition:

Subclass Of: <– section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.6.24. 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](#).

## 11.7. 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](#).

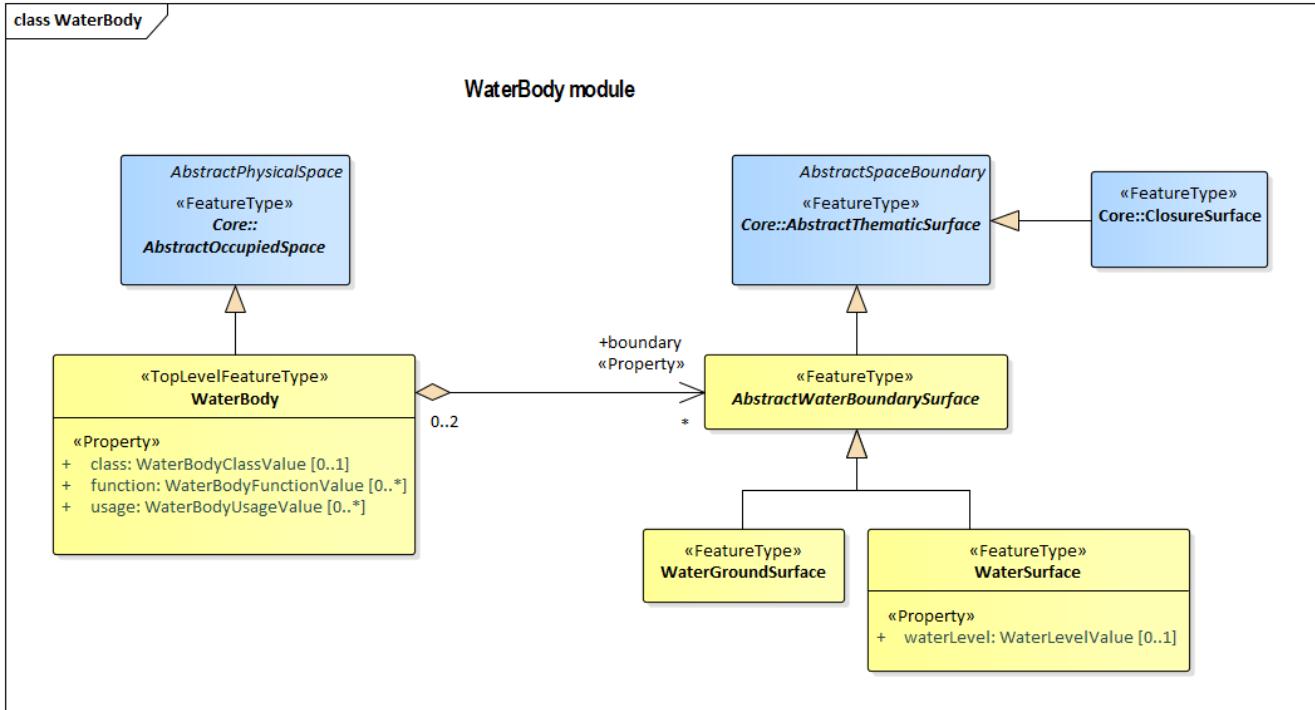


Figure 20. UML diagram of the Water Body Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

### 11.7.1. WaterBody Package

#### Package WaterBody

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.7.2. Class AbstractWaterBoundarySurface

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 <a href="#">Waterbody Data Dictionary</a> .

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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .

### Class AbstractWaterBoundarySurface

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.7.3. Class WaterBody

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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .

### Class WaterBody

Definition:

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

<b>Roles</b>	
Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">AbstractWaterBoundarySurface</a>
Definition:	
<b>Attributes</b>	
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»

#### 11.7.4. Class WaterBodyClassValue

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 <a href="#">Waterbody Data Dictionary</a> .

<b>Class WaterBodyClassValue</b>	
Definition:	
Subclass Of:	<-- section,>
Stereotype:	«CodeList»
<b>Roles</b>	
<b>Attributes</b>	

### 11.7.5. Class WaterBodyFunctionValue

<b>Requirement 143</b>	/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 <a href="#">Waterbody Data Dictionary</a> .

#### Class WaterBodyFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

### 11.7.6. Class WaterBodyUsageValue

<b>Requirement 144</b>	/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 <a href="#">Waterbody Data Dictionary</a> .

#### Class WaterBodyUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

### 11.7.7. Class WaterGroundSurface

<b>Requirement 145</b>	/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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .

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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .

### Class WaterGroundSurface

Definition:

Subclass Of: [AbstractWaterBoundarySurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.7.8. Class WaterLevelValue

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 <a href="#">Waterbody Data Dictionary</a> .

### Class WaterLevelValue

Definition:

Subclass Of: <-> section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.7.9. Class WaterSurface

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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .

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 <a href="#">Waterbody Data Dictionary</a> .
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 <a href="#">Waterbody Data Dictionary</a> .

## Class WaterSurface

Definition:

Subclass Of: [AbstractWaterBoundarySurface](#)

Stereotype: «FeatureType»

### Roles

### Attributes

Attribute Name: waterLevel

Value Type: WaterLevelValue

Definition: SIG3D: Codelist of the WaterSurface property waterLevel.

Multiplicity: [0..1]

Stereotype: «Property»

## 11.7.10. 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](#).

## 11.8. 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](#).

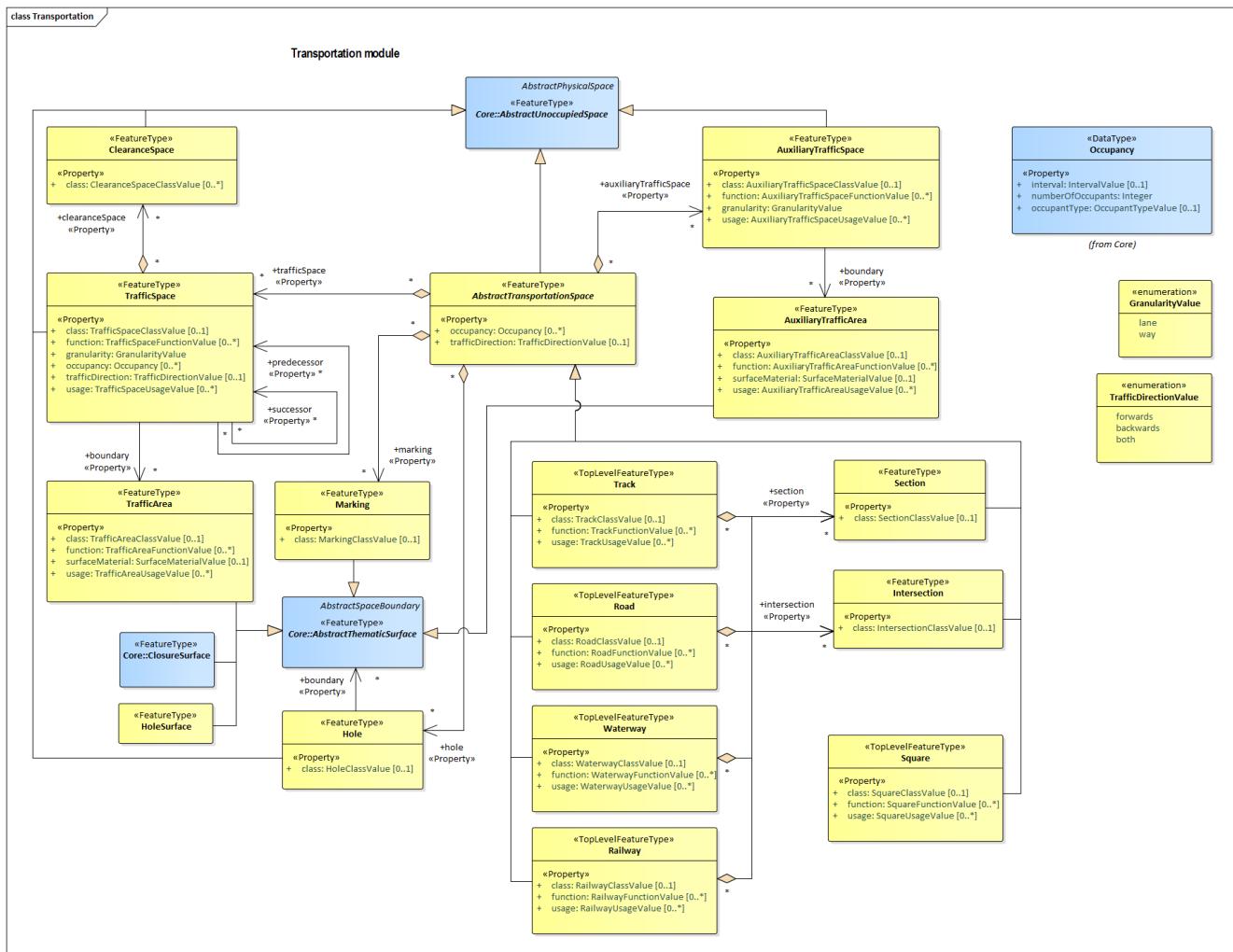


Figure 21. UML diagram of the Transportation Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.8.1. Transportation Package

### Package Transportation

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.8.2. Class AbstractTransportationSpace

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

## Class AbstractTransportationSpace

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

Role Name: trafficSpace

Cardinality: \*

Target Class: [TrafficSpace](#)

Definition: SIG3D: Relation between TransportationComplex and TrafficSpace

Role Name: hole

Cardinality: \*

Target Class: [Hole](#)

Definition:

Role Name: auxiliaryTrafficSpace

Cardinality: \*

Target Class: [AuxiliaryTrafficSpace](#)

Definition: SIG3D: Relation between TransportationComplex and AuxiliaryTrafficSpace.

Role Name: marking

Cardinality: \*

Target Class: [Marking](#)

Definition:

### Attributes

Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	trafficDirection
Value Type:	TrafficDirectionValue
Definition:	
Multiplicity:	[0..1]
Stereotype:	«Property»

### 11.8.3. Class AuxiliaryTrafficArea

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

<b>Class AuxiliaryTrafficArea</b>
Definition:
Subclass Of: <a href="#">AbstractThematicSurface</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>

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»

#### 11.8.4. Class AuxiliaryTrafficAreaClassValue

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 <a href="#">Transportation Data Dictionary</a> .

#### Class AuxiliaryTrafficAreaClassValue

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

#### Roles

#### Attributes

#### 11.8.5. Class AuxiliaryTrafficAreaFunctionValue

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 <a href="#">Transportation Data Dictionary</a> .
---	--

#### Class AuxiliaryTrafficAreaFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.6. Class AuxiliaryTrafficAreaUsageValue

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 <a href="#">Transportation Data Dictionary</a> .

#### Class AuxiliaryTrafficAreaUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.7. Class AuxiliaryTrafficSpace

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

## Class AuxiliaryTrafficSpace

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

Role Name: boundary

Cardinality: \*

Target Class: [AuxiliaryTrafficArea](#)

Definition:

### 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»

## 11.8.8. Class AuxiliaryTrafficSpaceClassValue

<b>Requirement 154</b>	/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 <a href="#">Transportation Data Dictionary</a> .

### Class AuxiliaryTrafficSpaceClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.8.9. Class AuxiliaryTrafficSpaceFunctionValue

<b>Requirement 155</b>	/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 <a href="#">Transportation Data Dictionary</a> .

### Class AuxiliaryTrafficSpaceFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.8.10. Class AuxiliaryTrafficSpaceUsageValue

<b>Requirement 156</b>	/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 <a href="#">Transportation Data Dictionary</a> .

## Class AuxiliaryTrafficSpaceUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.8.11. Class ClearanceSpace

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the ClearanceSpace UML class; including the name, definition, type, and cardinality of those documented in the ClearanceSpace section of the <a href="#">Transportation Data Dictionary</a> .

## Class ClearanceSpace

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

### Attributes

Attribute Name: class

Value Type: ClearanceSpaceClassValue

Definition:

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.8.12. Class ClearanceSpaceClassValue

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 <a href="#">Transportation Data Dictionary</a> .

### Class ClearanceSpaceClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.8.13. Class Hole

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

### Class Hole

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

#### Roles

Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">AbstractThematicSurface</a>
Definition:	

#### Attributes

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

### 11.8.14. Class HoleClassValue

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 <a href="#">Transportation Data Dictionary</a> .

#### Class HoleClassValue

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

#### Roles

#### Attributes

### 11.8.15. Class HoleSurface

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

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 <a href="#">Transportation Data Dictionary</a> .
---	--

### Class HoleSurface

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.8.16. Class Intersection

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

### Class Intersection

Definition:

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

Attribute Name: class

Value Type: IntersectionClassValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

### 11.8.17. Class IntersectionClassValue

<b>Requirement 163</b>	/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 <a href="#">Transportation Data Dictionary</a> .

#### Class IntersectionClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.18. Class Marking

<b>Requirement 164</b>	/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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

#### Class Marking

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

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

### 11.8.19. Class MarkingClassValue

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 <a href="#">Transportation Data Dictionary</a> .

#### Class MarkingClassValue

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

#### Roles

#### Attributes

### 11.8.20. Class Railway

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

#### Class Railway

Definition:	
Subclass Of:	<a href="#">AbstractTransportationSpace</a>
Stereotype:	«TopLevelFeatureType»
<b>Roles</b>	
Role Name:	section
Cardinality:	*
Target Class:	<a href="#">Section</a>
Definition:	
Role Name:	intersection
Cardinality:	*
Target Class:	<a href="#">Intersection</a>
Definition:	
<b>Attributes</b>	
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»

## 11.8.21. Class RailwayClassValue

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 <a href="#">Transportation Data Dictionary</a> .

<b>Class RailwayClassValue</b>	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«CodeList»
<b>Roles</b>	

## Attributes

### 11.8.22. Class RailwayFunctionValue

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 <a href="#">Transportation Data Dictionary</a> .

## Class RailwayFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

## Roles

## Attributes

### 11.8.23. Class RailwayUsageValue

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 <a href="#">Transportation Data Dictionary</a> .

## Class RailwayUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

## Roles

## Attributes

### 11.8.24. Class Road

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 <a href="#">Transportation Data Dictionary</a> .

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

## Class Road

Definition:

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

### Roles

Role Name: intersection

Cardinality: \*

Target Class: [Intersection](#)

Definition:

Role Name: section

Cardinality: \*

Target Class: [Section](#)

Definition:

### 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»

### 11.8.25. Class RoadClassValue

<b>Requirement 171</b>	/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 <a href="#">Transportation Data Dictionary</a> .

#### Class RoadClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.26. Class RoadFunctionValue

<b>Requirement 172</b>	/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 <a href="#">Transportation Data Dictionary</a> .

#### Class RoadFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.27. Class RoadUsageValue

<b>Requirement 173</b>	/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 <a href="#">Transportation Data Dictionary</a> .

#### Class RoadUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

<b>Roles</b>	
<b>Attributes</b>	

### 11.8.28. Class Section

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

<b>Class Section</b>
Definition:
Subclass Of: <a href="#">AbstractTransportationSpace</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>
Attribute Name: class
Value Type: SectionClassValue
Definition:
Multiplicity: [0..1]
Stereotype: «Property»

### 11.8.29. Class SectionClassValue

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 <a href="#">Transportation Data Dictionary</a> .

<b>Class Section</b>	<b>ClassValue</b>
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«CodeList»
<b>Roles</b>	
<b>Attributes</b>	

### 11.8.30. Class Square

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

<b>Class Square</b>
Definition:
Subclass Of: <a href="#">AbstractTransportationSpace</a>
Stereotype: «TopLevelFeatureType»
<b>Roles</b>
<b>Attributes</b>
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»

### 11.8.31. Class SquareClassValue

<b>Requirement 177</b>	/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 <a href="#">Transportation Data Dictionary</a> .

#### Class SquareClassValue

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

#### Roles

#### Attributes

### 11.8.32. Class SquareFunctionValue

<b>Requirement 178</b>	/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 <a href="#">Transportation Data Dictionary</a> .

#### Class SquareFunctionValue

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

#### Roles

#### Attributes

### 11.8.33. Class SquareUsageValue

<b>Requirement 179</b>	/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 <a href="#">Transportation Data Dictionary</a> .
---	---

### Class SquareUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.34. Class SurfaceMaterialValue

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 <a href="#">Transportation Data Dictionary</a> .

### Class SurfaceMaterialValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.35. Class Track

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

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 <a href="#">Transportation Data Dictionary</a> .
---	--

## Class Track

Definition:

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

### Roles

Role Name: section

Cardinality: \*

Target Class: [Section](#)

Definition:

Role Name: intersection

Cardinality: \*

Target Class: [Intersection](#)

Definition:

### 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»

## 11.8.36. Class TrackClassValue

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 <a href="#">Transportation Data Dictionary</a> .

## Class TrackClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.8.37. Class TrackFunctionValue

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 <a href="#">Transportation Data Dictionary</a> .

## Class TrackFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.8.38. Class TrackUsageValue

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 <a href="#">Transportation Data Dictionary</a> .

## Class TrackUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.8.39. Class TrafficArea

<b>Requirement 185</b>	/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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

## Class TrafficArea

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

### Roles

### 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»

## 11.8.40. Class TrafficAreaClassValue

<b>Requirement 186</b>	/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 TrafficAreaClassValue section of the <a href="#">Transportation Data Dictionary</a> .

### Class TrafficAreaClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.8.41. Class TrafficAreaFunctionValue

<b>Requirement 187</b>	/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 <a href="#">Transportation Data Dictionary</a> .

### Class TrafficAreaFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.8.42. Class TrafficAreaUsageValue

<b>Requirement 188</b>	/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 <a href="#">Transportation Data Dictionary</a> .

## Class TrafficAreaUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.8.43. Class TrafficSpace

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

## Class TrafficSpace

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

Role Name: clearanceSpace

Cardinality: \*

Target Class: [ClearanceSpace](#)

Definition:

Role Name: successor

Cardinality: \*

Target Class: [TrafficSpace](#)

Definition:

Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">TrafficArea</a>
Definition:	
Role Name:	predecessor
Cardinality:	*
Target Class:	<a href="#">TrafficSpace</a>
Definition:	
<b>Attributes</b>	
Attribute Name:	class
Value Type:	TrafficSpaceClassValue
Definition:	SIG3D: Classification of TrafficSpace as given by the relevant national regulations, information communities, or specific applications.
Multiplicity:	[0..1]
Stereotype:	«Property»
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»

#### 11.8.44. Class TrafficSpaceClassValue

<b>Requirement 190</b>	/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 TrafficSpaceClassValue section of the <a href="#">Transportation Data Dictionary</a> .

##### Class TrafficSpaceClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.8.45. Class TrafficSpaceFunctionValue

<b>Requirement 191</b>	/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 <a href="#">Transportation Data Dictionary</a> .

##### Class TrafficSpaceFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.8.46. Class TrafficSpaceUsageValue

<b>Requirement 192</b>	/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 <a href="#">Transportation Data Dictionary</a> .

##### Class TrafficSpaceUsageValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

#### 11.8.47. Class TransportationSpaceClassValue

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 <a href="#">Transportation Data Dictionary</a> .

<b>Class TransportationSpaceClassValue</b>
Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

#### 11.8.48. Class TransportationSpaceFunctionValue

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 <a href="#">Transportation Data Dictionary</a> .

<b>Class TransportationSpaceFunctionValue</b>
Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

#### 11.8.49. Class TransportationSpaceUsageValue

Requirement 195	/req/Transportation/TransportationSpaceUsageValue
<hr/>	

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 <a href="#">Transportation Data Dictionary</a> .
---	--

### Class TransportationSpaceUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.8.50. Class Waterway

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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .
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 <a href="#">Transportation Data Dictionary</a> .

### Class Waterway

Definition:

Subclass Of: [AbstractTransportationSpace](#)

Stereotype: «TopLevelFeatureType»

#### Roles

Role Name: section

Cardinality: \*

Target Class: [Section](#)

Definition:

Role Name: intersection

Cardinality: \*

Target Class: [Intersection](#)

Definition:

### Attributes

Attribute Name: class

Value Type: WaterwayClassValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: WaterwayFunctionValue

Definition:

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: usage

Value Type: WaterwayUsageValue

Definition:

Multiplicity: [0..\*]

Stereotype: «Property»

## 11.8.51. Class WaterwayClassValue

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 <a href="#">Transportation Data Dictionary</a> .

### Class WaterwayClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.8.52. Class WaterwayFunctionValue

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 <a href="#">Transportation Data Dictionary</a> .
---	--

#### Class WaterwayFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.53. Class WaterwayUsageValue

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 <a href="#">Transportation Data Dictionary</a> .

#### Class WaterwayUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.8.54. Class GranularityValue

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 <a href="#">Transportation Data Dictionary</a> .

#### Class GranularityValue

Definition:

Subclass Of: <-- section,>>

Stereotype:

<b>Roles</b>	
<b>Attributes</b>	
Attribute Name:	lane
Value Type:	
Definition:	
Multiplicity:	
Stereotype:	
Attribute Name:	way
Value Type:	
Definition:	
Multiplicity:	
Stereotype:	

### 11.8.55. Class TrafficDirectionValue

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 <a href="#">Transportation Data Dictionary</a> .

<b>Class TrafficDirectionValue</b>	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	
<b>Roles</b>	
<b>Attributes</b>	
Attribute Name:	forwards
Value Type:	
Definition:	
Multiplicity:	
Stereotype:	
Attribute Name:	backwards
Value Type:	
Definition:	
Multiplicity:	
Stereotype:	

Attribute Name: both

Value Type:

Definition:

Multiplicity:

Stereotype:

### 11.8.56. 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](#).

## 11.9. 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](#).

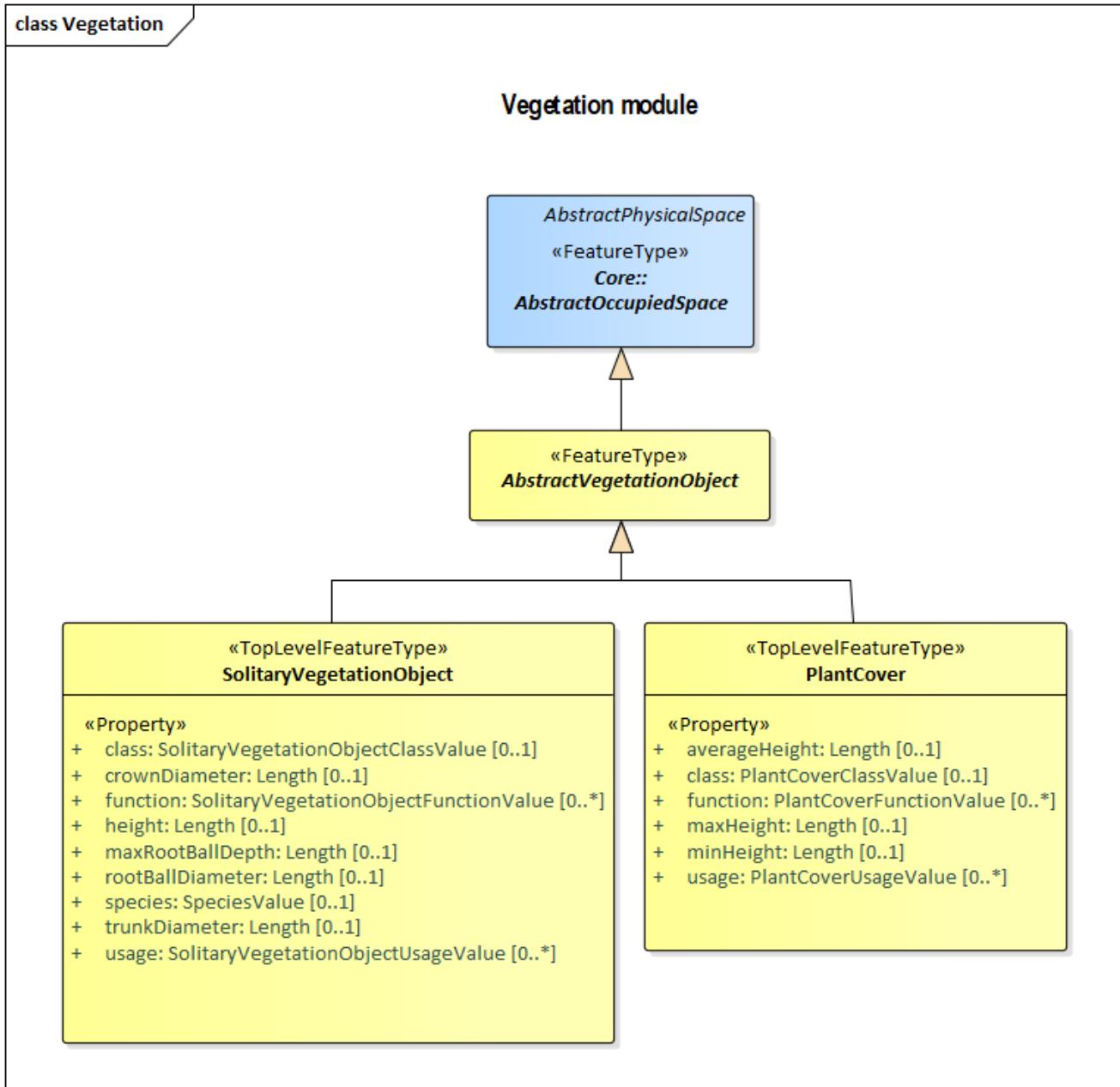


Figure 22. UML diagram of the Vegetation Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

### 11.9.1. Vegetation Package

#### Package Vegetation

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.9.2. Class AbstractVegetationObject

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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .

### Class AbstractVegetationObject

Definition:

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

## 11.9.3. Class PlantCover

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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .

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 <a href="#">Vegetation Data Dictionary</a> .
---	--

## Class PlantCover

Definition:

Subclass Of: [AbstractVegetationObject](#)

Stereotype: «TopLevelFeatureType»

## Roles

### 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»

#### 11.9.4. Class PlantCoverClassValue

<b>Requirement 204</b>	/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 <a href="#">Vegetation Data Dictionary</a> .

##### Class PlantCoverClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.9.5. Class PlantCoverFunctionValue

<b>Requirement 205</b>	/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 <a href="#">Vegetation Data Dictionary</a> .

##### Class PlantCoverFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.9.6. Class PlantCoverUsageValue

<b>Requirement 206</b>	/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 <a href="#">Vegetation Data Dictionary</a> .

##### Class PlantCoverUsageValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

## 11.9.7. Class SolitaryVegetationObject

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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .
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 <a href="#">Vegetation Data Dictionary</a> .

<b>Class SolitaryVegetationObject</b>
Definition:
Subclass Of: <a href="#">AbstractVegetationObject</a>
Stereotype: «TopLevelFeatureType»
<b>Roles</b>
<b>Attributes</b>
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»

## 11.9.8. Class SolitaryVegetationObjectClassValue

<b>Requirement 208</b>	/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 <a href="#">Vegetation Data Dictionary</a> .

### Class SolitaryVegetationObjectClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

## 11.9.9. Class SolitaryVegetationObjectFunctionValue

<b>Requirement 209</b>	/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 <a href="#">Vegetation Data Dictionary</a> .

### Class SolitaryVegetationObjectFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

Roles

Attributes

## 11.9.10. Class SolitaryVegetationObjectUsageValue

<b>Requirement 210</b>	/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 <a href="#">Vegetation Data Dictionary</a> .

## Class SolitaryVegetationObjectUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Roles**

**Attributes**

## 11.9.11. Class SpeciesValue

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 <a href="#">Vegetation Data Dictionary</a> .

## Class SpeciesValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

**Roles**

**Attributes**

## 11.9.12. 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](#).

## 11.10. 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](#)

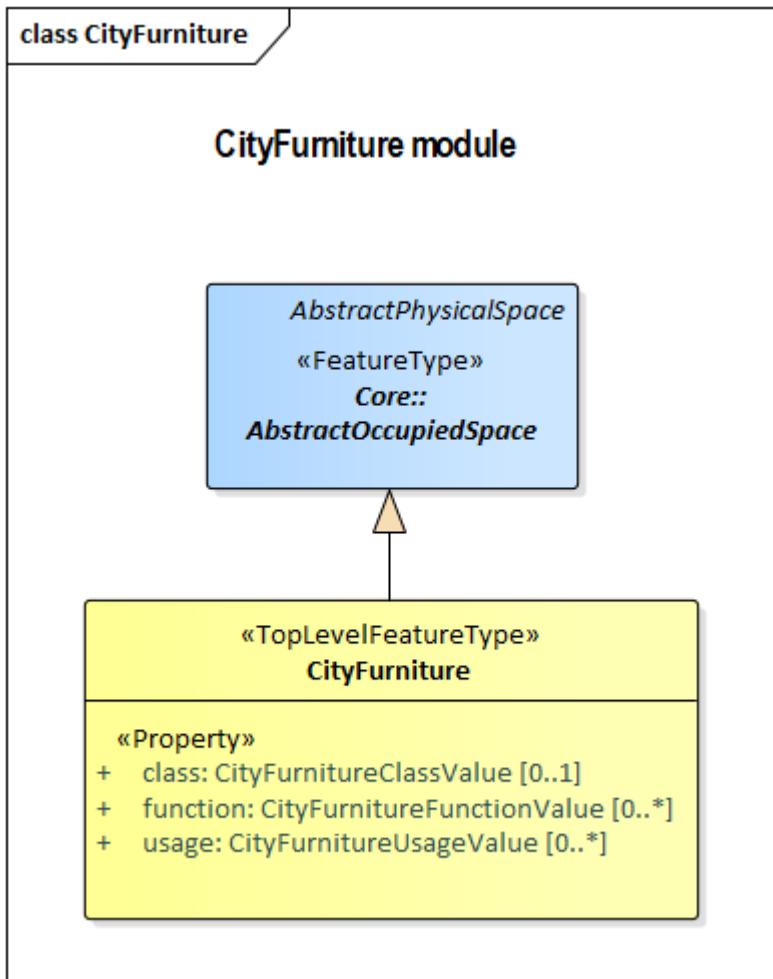


Figure 23. UML diagram of CityGML's city furniture model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

### 11.10.1. CityFurniture Package

#### Package CityFurniture

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.10.2. Class CityFurniture

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 <a href="#">CityFurniture Data Dictionary</a> .
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 <a href="#">CityFurniture Data Dictionary</a> .
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 <a href="#">CityFurniture Data Dictionary</a> .
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 <a href="#">CityFurniture Data Dictionary</a> .

## Class CityFurniture

Definition:

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

### Roles

### 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»

### 11.10.3. Class CityFurnitureClassValue

<b>Requirement 213</b>	/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 CityFurnitureClassValue section of the <a href="#">CityFurniture Data Dictionary</a> .

#### Class CityFurnitureClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.10.4. Class CityFurnitureFunctionValue

<b>Requirement 214</b>	/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 CityFurnitureFunctionValue section of the <a href="#">CityFurniture Data Dictionary</a> .

#### Class CityFurnitureFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.10.5. Class CityFurnitureUsageValue

<b>Requirement 215</b>	/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 <a href="#">CityFurniture Data Dictionary</a> .

#### Class CityFurnitureUsageValue

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

**Roles**

**Attributes**

## 11.10.6. 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](#).

## 11.11. 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](#).

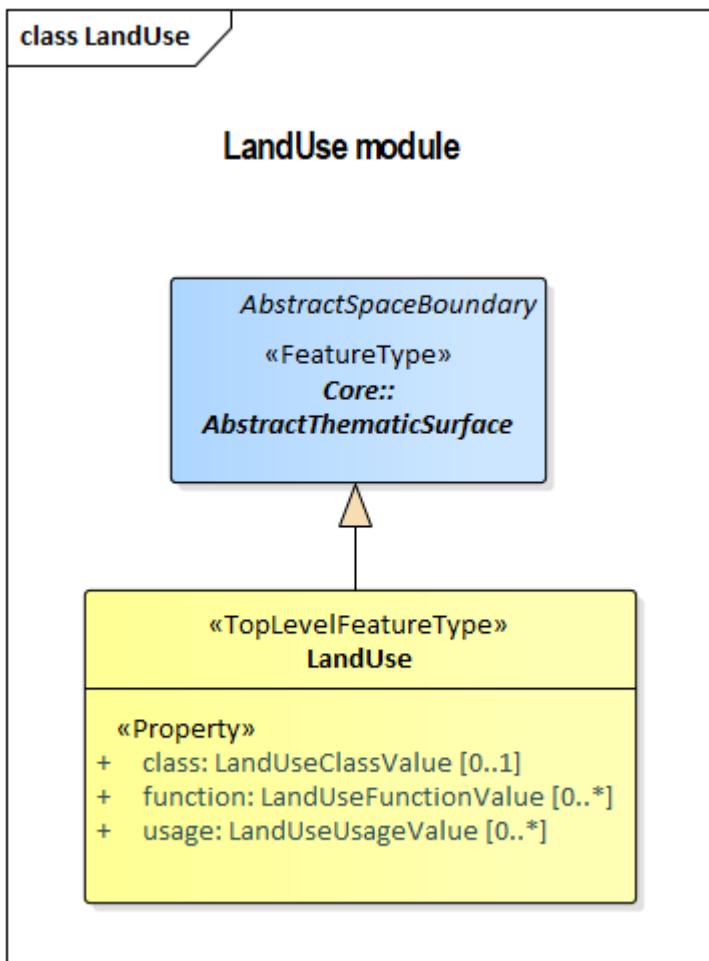


Figure 24. UML diagram of the Land Use Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

### 11.11.1. LandUse Package

#### Package LandUse

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.11.2. Class LandUse

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 <a href="#">LandUse Data Dictionary</a> .

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 <a href="#">LandUse Data Dictionary</a> .
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 <a href="#">LandUse Data Dictionary</a> .
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 <a href="#">LandUse Data Dictionary</a> .

### Class LandUse

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «TopLevelFeatureType»

#### Roles

#### 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»

### 11.11.3. Class LandUseClassValue

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 LandUseClassValue section of the <a href="#">LandUse Data Dictionary</a> .
---	---

### Class LandUseClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.11.4. Class LandUseFunctionValue

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 LandUseFunctionValue section of the <a href="#">LandUse Data Dictionary</a> .

### Class LandUseFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.11.5. Class LandUseUsageValue

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 <a href="#">LandUse Data Dictionary</a> .

### Class LandUseUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.11.6. 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](#).

## 11.12. 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](#).

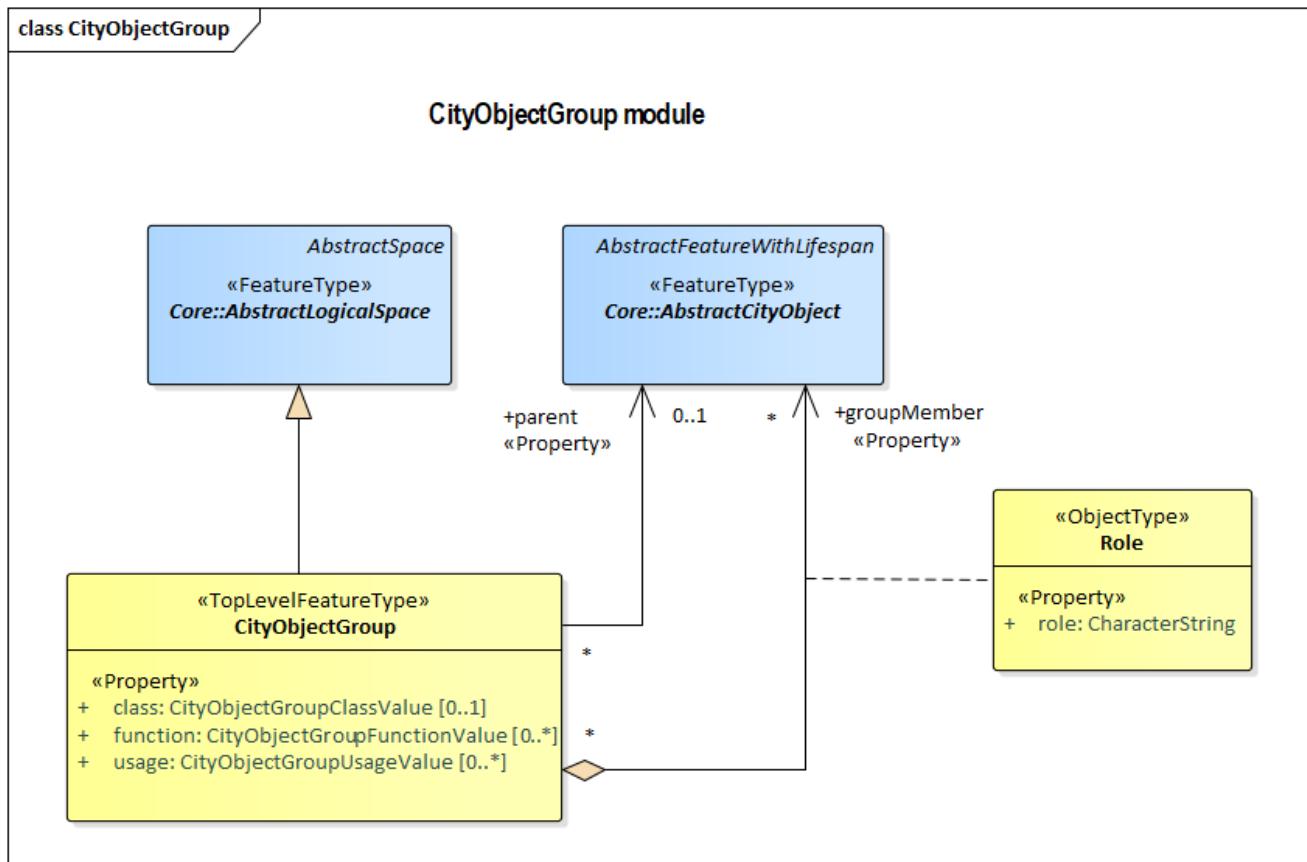


Figure 25. UML diagram of the City Object Group Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.12.1. CityObjectGroup Package

### Package CityObjectGroup

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.12.2. Class CityObjectGroup

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 <a href="#">CityObjectGroup Data Dictionary</a> .
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 <a href="#">CityObjectGroup Data Dictionary</a> .
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 <a href="#">CityObjectGroup Data Dictionary</a> .
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 <a href="#">CityObjectGroup Data Dictionary</a> .

### Class CityObjectGroup

Definition:

Subclass Of: [AbstractLogicalSpace](#)

Stereotype: «TopLevelFeatureType»

### Roles

Role Name: parent

Cardinality: 0..1

Target Class: [AbstractCityObject](#)

Definition: SIG3D: Reference to an AbstractCityObject related to the aggregation as a whole (e.g. the corresponding building object for the aggregation of rooms to a storey).

Role Name:	groupMember
Cardinality:	*
Target Class:	<a href="#">AbstractCityObject</a>
Definition:	SIG3D: Reference to a city object.

### 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»

## 11.12.3. Class CityObjectGroupClassValue

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 <a href="#">CityObjectGroup Data Dictionary</a> .

### Class CityObjectGroupClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

#### 11.12.4. Class CityObjectGroupFunctionValue

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 <a href="#">CityObjectGroup Data Dictionary</a> .

##### Class CityObjectGroupFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.12.5. Class CityObjectGroupUsageValue

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 <a href="#">CityObjectGroup Data Dictionary</a> .

##### Class CityObjectGroupUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

##### Roles

##### Attributes

#### 11.12.6. Class Role

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 <a href="#">CityObjectGroup Data Dictionary</a> .
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 <a href="#">CityObjectGroup Data Dictionary</a> .

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 <a href="#">CityObjectGroup Data Dictionary</a> .
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 <a href="#">CityObjectGroup Data Dictionary</a> .

## Class Role

Definition:

Subclass Of: <-- section,>>

Stereotype: «ObjectType»

### Roles

#### Attributes

Attribute Name: role

Value Type: CharacterString

Definition: SIG3D: Description of the role.

Multiplicity:

Stereotype: «Property»

## 11.12.7. 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](#).

## 11.13. 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](#).

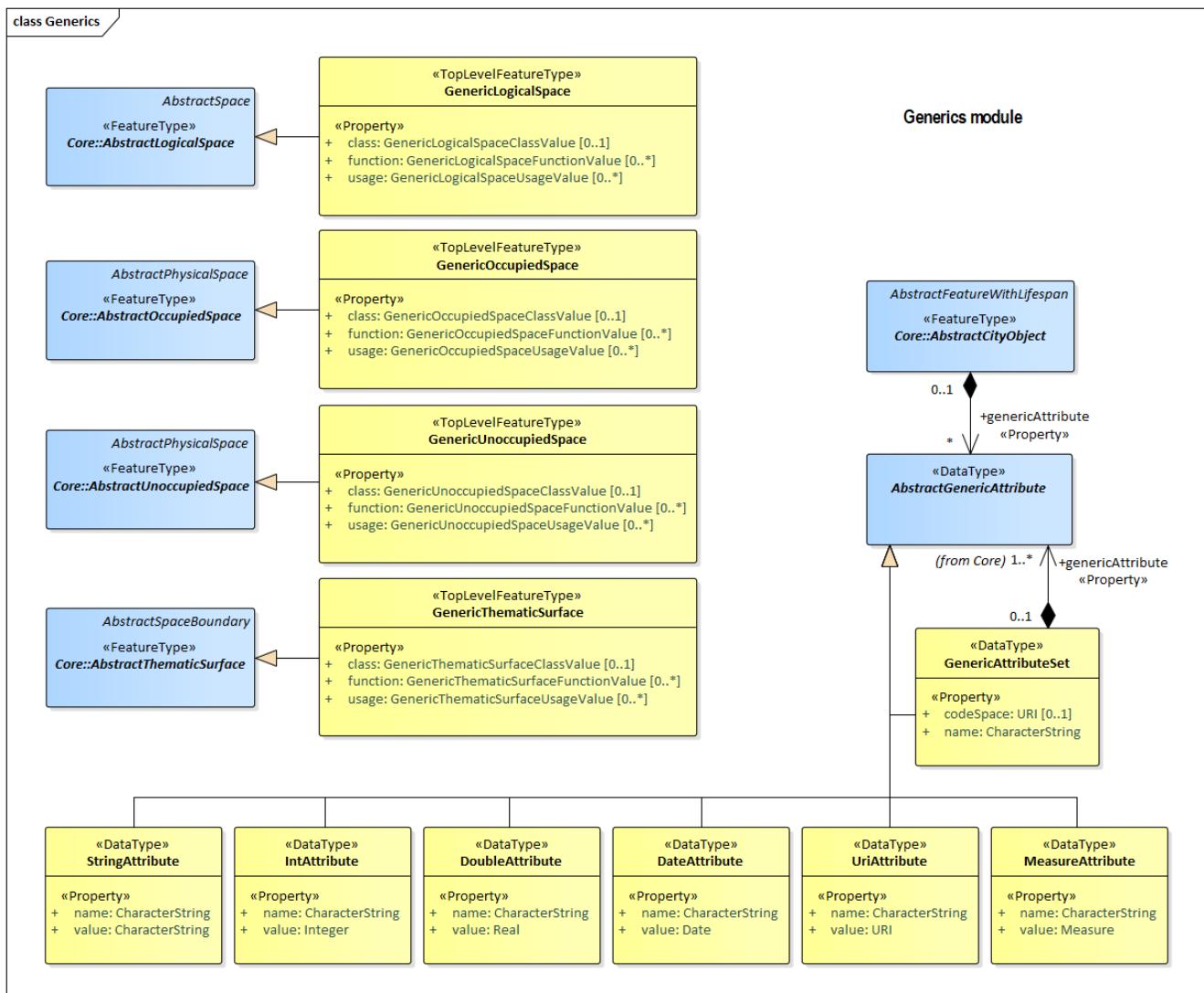


Figure 26. UML diagram of the Generics Model.

The UML diagram of the Generics Model. is color coded as follows:

Yellow	indicates
Blue	indicates
Pink	indicates

### 11.13.1. Generics Package

#### Package Generics

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

### 11.13.2. Class GenericLogicalSpace

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

### Class GenericLogicalSpace

Definition:

Subclass Of: [AbstractLogicalSpace](#)

Stereotype: «TopLevelFeatureType»

#### Roles

#### 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»

### 11.13.3. Class GenericLogicalSpaceClassValue

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 <a href="#">Generics Data Dictionary</a> .
---	--

#### Class GenericLogicalSpaceClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.13.4. Class GenericLogicalSpaceFunctionValue

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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericLogicalSpaceFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.13.5. Class GenericLogicalSpaceUsageValue

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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericLogicalSpaceUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

<b>Roles</b>
<b>Attributes</b>

### 11.13.6. Class GenericOccupiedSpace

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericOccupiedSpace

Definition:

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

#### Roles

#### Attributes

Attribute Name: class

Value Type: GenericOccupiedSpaceClassValue

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: function

Value Type: GenericOccupiedSpaceFunctionValue

Definition:

Multiplicity: [0..\*]

Stereotype: «Property»

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

### 11.13.7. Class GenericOccupiedSpaceClassValue

<b>Requirement 230</b>	/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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericOccupiedSpaceClassValue

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

#### Roles

#### Attributes

### 11.13.8. Class GenericOccupiedSpaceFunctionValue

<b>Requirement 231</b>	/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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericOccupiedSpaceFunctionValue

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

#### Roles

#### Attributes

### 11.13.9. Class GenericOccupiedSpaceUsageValue

<b>Requirement 232</b>	/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 <a href="#">Generics Data Dictionary</a> .
---	---

### Class GenericOccupiedSpaceUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.13.10. Class GenericThematicSurface

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

### Class GenericThematicSurface

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «TopLevelFeatureType»

#### Roles

#### 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»

### 11.13.11. Class GenericThematicSurfaceClassValue

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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericThematicSurfaceClassValue

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

#### Roles

#### Attributes

### 11.13.12. Class GenericThematicSurfaceFunctionValue

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 <a href="#">Generics Data Dictionary</a> .

#### Class GenericThematicSurfaceFunctionValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.13.13. Class GenericThematicSurfaceUsageValue

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 <a href="#">Generics Data Dictionary</a> .

<b>Class GenericThematicSurfaceUsageValue</b>
Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.13.14. Class GenericUnoccupiedSpace

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

## Class GenericUnoccupiedSpace

Definition:

Subclass Of: [AbstractUnoccupiedSpace](#)

Stereotype: «TopLevelFeatureType»

### Roles

### 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»

## 11.13.15. Class GenericUnoccupiedSpaceClassValue

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 <a href="#">Generics Data Dictionary</a> .

## Class GenericUnoccupiedSpaceClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.13.16. Class GenericUnoccupiedSpaceFunctionValue

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 <a href="#">Generics Data Dictionary</a> .
---	--

### Class GenericUnoccupiedSpaceFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.13.17. Class GenericUnoccupiedSpaceUsageValue

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 <a href="#">Generics Data Dictionary</a> .

### Class GenericUnoccupiedSpaceUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.13.18. Class DateAttribute

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

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 <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the DateAttribute UML class; including the name, definition, type, and cardinality of those documented in the DateAttribute section of the <a href="#">Generics Data Dictionary</a> .

### Class DateAttribute

Definition:

Subclass Of: <-> section,>>

Stereotype: «DataType»

#### Roles

#### 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»

### 11.13.19. Class DoubleAttribute

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

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 <a href="#">Generics Data Dictionary</a> .
---	--

### Class DoubleAttribute

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

#### 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»

### 11.13.20. Class GenericAttributeSet

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

## Class GenericAttributeSet

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Roles

Role Name: genericAttribute

Cardinality: 1..\*

Target Class: [AbstractGenericAttribute](#)

Definition:

### 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»

## 11.13.21. Class IntAttribute

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

## Class IntAttribute

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

### Roles

### 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»

## 11.13.22. Class MeasureAttribute

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

### Class MeasureAttribute

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

### Roles

<b>Attributes</b>	
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»

### 11.13.23. Class StringAttribute

<b>Requirement 246</b>	/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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .

<b>Class StringAttribute</b>
Definition:
Subclass Of: <-> section,>>
Stereotype: «DataType»
<b>Roles</b>
<b>Attributes</b>

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»

### 11.13.24. Class UriAttribute

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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
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 <a href="#">Generics Data Dictionary</a> .
D	The implementation Specification SHALL represent the attributes of all parent classes of the UriAttribute UML class; including the name, definition, type, and cardinality of those documented in the UriAttribute section of the <a href="#">Generics Data Dictionary</a> .

#### Class UriAttribute

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

#### Roles

#### 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»

### 11.13.25. 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](#).

## 11.14. Construction

Requirements Class	
<a href="http://www.opengis.net/spec/CityGML/3.1/req/req-class-construction">http://www.opengis.net/spec/CityGML/3.1/req/req-class-construction</a>	
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](#).

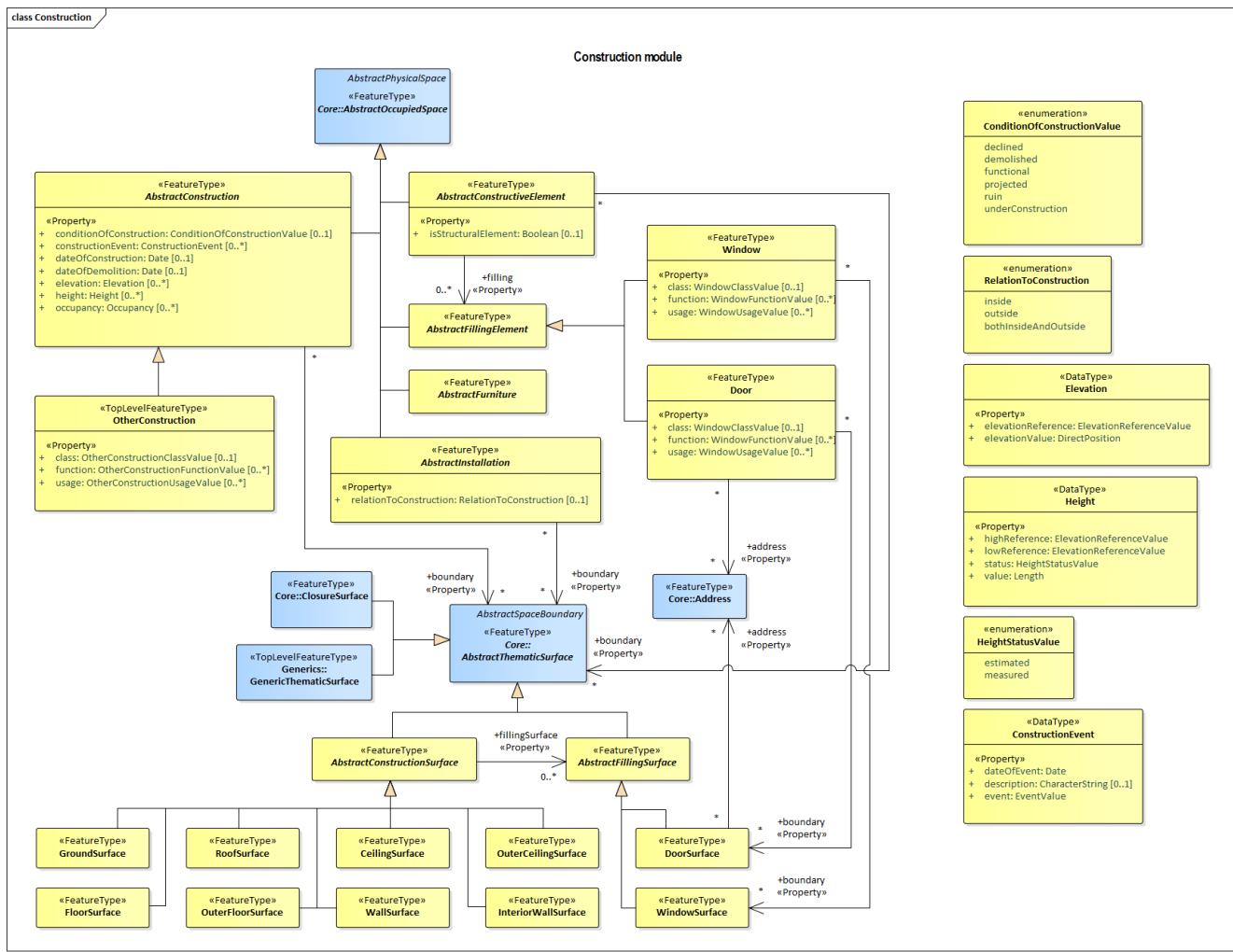


Figure 27. UML diagram of the Construction Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.14.1. Construction Package

### Package Construction

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.14.2. Class AbstractConstruction

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

## Class AbstractConstruction

Definition:

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

Role Name: boundary

Cardinality: \*

Target Class: [AbstractThematicSurface](#)

Definition:

### Attributes

Attribute Name: conditionOfConstruction

Value Type: ConditionOfConstructionValue

Definition: [INSPIRE] Status of the construction.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: constructionEvent

Value Type: ConstructionEvent

Definition: Date of renovation of a construction.

Multiplicity: [0..\*]

Stereotype: «Property»

Attribute Name: dateOfConstruction

Value Type: Date

Definition: Date of completion of a construction.

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name:	dateOfDemolition
Value Type:	Date
Definition:	Date of demolition of a construction.
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:	height
Value Type:	Height
Definition:	[INSPIRE] Height above ground.
Multiplicity:	[0..*]
Stereotype:	«Property»
Attribute Name:	occupancy
Value Type:	Occupancy
Definition:	
Multiplicity:	[0..*]
Stereotype:	«Property»

### 11.14.3. Class AbstractConstructionSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

## Class AbstractConstructionSurface

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

### Roles

Role Name: fillingSurface

Cardinality: 0..\*

Target Class: [AbstractFillingSurface](#)

Definition:

### Attributes

## 11.14.4. Class AbstractConstructiveElement

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

## Class AbstractConstructiveElement

Definition:

Subclass Of: [AbstractOccupiedSpace](#)

Stereotype: «FeatureType»

### Roles

Role Name: filling

Cardinality: 0..\*

Target Class: [AbstractFillingElement](#)

Definition:

Role Name:	boundary
Cardinality:	*
Target Class:	<a href="#">AbstractThematicSurface</a>
Definition:	

#### Attributes

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

### 11.14.5. Class AbstractFillingElement

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class AbstractFillingElement

Definition:	
Subclass Of:	<a href="#">AbstractOccupiedSpace</a>
Stereotype:	«FeatureType»

#### Roles

#### Attributes

### 11.14.6. Class AbstractFillingSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class AbstractFillingSurface

Definition:

Subclass Of: [AbstractThematicSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.14.7. Class AbstractFurniture

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class AbstractFurniture

Definition:
Subclass Of: <a href="#">AbstractOccupiedSpace</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>

### 11.14.8. Class AbstractInstallation

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

Class AbstractInstallation
Definition:
Subclass Of: <a href="#">AbstractOccupiedSpace</a>
Stereotype: «FeatureType»
<b>Roles</b>
Role Name: boundary
Cardinality: *
Target Class: <a href="#">AbstractThematicSurface</a>
Definition:
<b>Attributes</b>
Attribute Name: relationToConstruction
Value Type: RelationToConstruction
Definition:
Multiplicity: [0..1]
Stereotype: «Property»

## 11.14.9. Class CeilingSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

### Class CeilingSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

## 11.14.10. Class Door

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

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 <a href="#">Construction Data Dictionary</a> .
---	--

## Class Door

Definition:

Subclass Of: [AbstractFillingElement](#)

Stereotype: «FeatureType»

### Roles

Role Name: boundary

Cardinality: \*

Target Class: [DoorSurface](#)

Definition:

Role Name: address

Cardinality: \*

Target Class: [Address](#)

Definition: SIG3D: Relation between Door and Address.

### 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»

## 11.14.11. Class DoorClassName

Requirement 257	/req/Construction/DoorClassName
A	Any use of the DoorClassName type in an Implementation Specification SHALL be restricted to the valid values specified in the DoorClassName UML class as documented in the DoorClassName section of the <a href="#">Construction Data Dictionary</a> .

## Class DoorClassValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.14.12. Class DoorFunctionValue

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 <a href="#">Construction Data Dictionary</a> .

## Class DoorFunctionValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.14.13. Class DoorSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

## Class DoorSurface

Definition:

Subclass Of: [AbstractFillingSurface](#)

Stereotype: «FeatureType»

### Roles

Role Name: address

Cardinality: \*

Target Class: [Address](#)

Definition:

### Attributes

## 11.14.14. Class DoorUsageValue

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 <a href="#">Construction Data Dictionary</a> .

## Class DoorUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.14.15. Class ElevationReferenceValue

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 <a href="#">Construction Data Dictionary</a> .

## Class ElevationReferenceValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

### Roles

### Attributes

## 11.14.16. Class EventValue

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 <a href="#">Construction Data Dictionary</a> .

### Class EventValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

## 11.14.17. Class FloorSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

### Class FloorSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

## 11.14.18. Class GroundSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

### Class GroundSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

## 11.14.19. Class InteriorWallSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

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 <a href="#">Construction Data Dictionary</a> .
---	--

### Class InteriorWallSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.14.20. Class OtherConstruction

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

### Class OtherConstruction

Definition:

Subclass Of: [AbstractConstruction](#)

Stereotype: «TopLevelFeatureType»

#### Roles

#### 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»

### 11.14.21. Class OtherConstructionClassValue

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 <a href="#">Construction Data Dictionary</a> .

Class OtherConstructionClassValue	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«CodeList»
Roles	
Attributes	

### 11.14.22. Class OtherConstructionFunctionValue

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 <a href="#">Construction Data Dictionary</a> .

Class OtherConstructionFunctionValue	

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

### 11.14.23. Class OtherConstructionUsageValue

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 <a href="#">Construction Data Dictionary</a> .

Class OtherConstructionUsageValue	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	«CodeList»
<b>Roles</b>	
<b>Attributes</b>	

### 11.14.24. Class OuterCeilingSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

## Class OuterCeilingSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

### Roles

### Attributes

## 11.14.25. Class OuterFloorSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

## Class OuterFloorSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

### Roles

### Attributes

## 11.14.26. Class RoofSurface

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 <a href="#">Construction Data Dictionary</a> .

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class RoofSurface

Definition:

Subclass Of: [AbstractConstructionSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.14.27. Class WallSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class WallSurface

Definition:
Subclass Of: <a href="#">AbstractConstructionSurface</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>

### 11.14.28. Class Window

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

<b>Class Window</b>
Definition:
Subclass Of: <a href="#">AbstractFillingElement</a>
Stereotype: «FeatureType»
<b>Roles</b>
Role Name: boundary
Cardinality: *
Target Class: <a href="#">WindowSurface</a>
Definition:
<b>Attributes</b>
Attribute Name: class
Value Type: WindowClassName
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»

## 11.14.29. Class WindowClassValue

Requirement 275	/req/Construction/WindowClassValue
A	Any use of the WindowClassValue type in an Implementation Specification SHALL be restricted to the valid values specified in the WindowClassValue UML class as documented in the <a href="#">Construction Data Dictionary</a> .

### Class WindowClassValue

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

#### Roles

#### Attributes

## 11.14.30. Class WindowFunctionValue

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 <a href="#">Construction Data Dictionary</a> .

### Class WindowFunctionValue

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

#### Roles

#### Attributes

### 11.14.31. Class WindowSurface

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class WindowSurface

Definition:

Subclass Of: [AbstractFillingSurface](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

### 11.14.32. Class WindowUsageValue

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 <a href="#">Construction Data Dictionary</a> .

#### Class WindowUsageValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.14.33. Class ConditionOfConstructionValue

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 <a href="#">Construction Data Dictionary</a> .

#### Class ConditionOfConstructionValue

Definition:

Subclass Of: <-- section,>>

Stereotype:

Roles

#### 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:

### 11.14.34. Class ConstructionEvent

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class ConstructionEvent

Definition:

Subclass Of: <-- section,>>

StereoType: «DataType»

#### Roles

#### 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»

### 11.14.35. Class Elevation

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class Elevation

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

#### Roles

#### 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»

### 11.14.36. Class Height

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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .
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 <a href="#">Construction Data Dictionary</a> .

#### Class Height

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

#### Roles

#### 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»

### 11.14.37. Class HeightStatusValue

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 <a href="#">Construction Data Dictionary</a> .

Class HeightStatusValue	
Definition:	
Subclass Of:	<-- section,>>
Stereotype:	
<b>Roles</b>	
<b>Attributes</b>	
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:	

### 11.14.38. Class RelationToConstruction

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 <a href="#">Construction Data Dictionary</a> .
---	---

### Class RelationToConstruction

Definition:

Subclass Of: <-- section,>>

Stereotype:

#### Roles

#### 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:

### 11.14.39. 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](#).

## 11.15. 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](#).

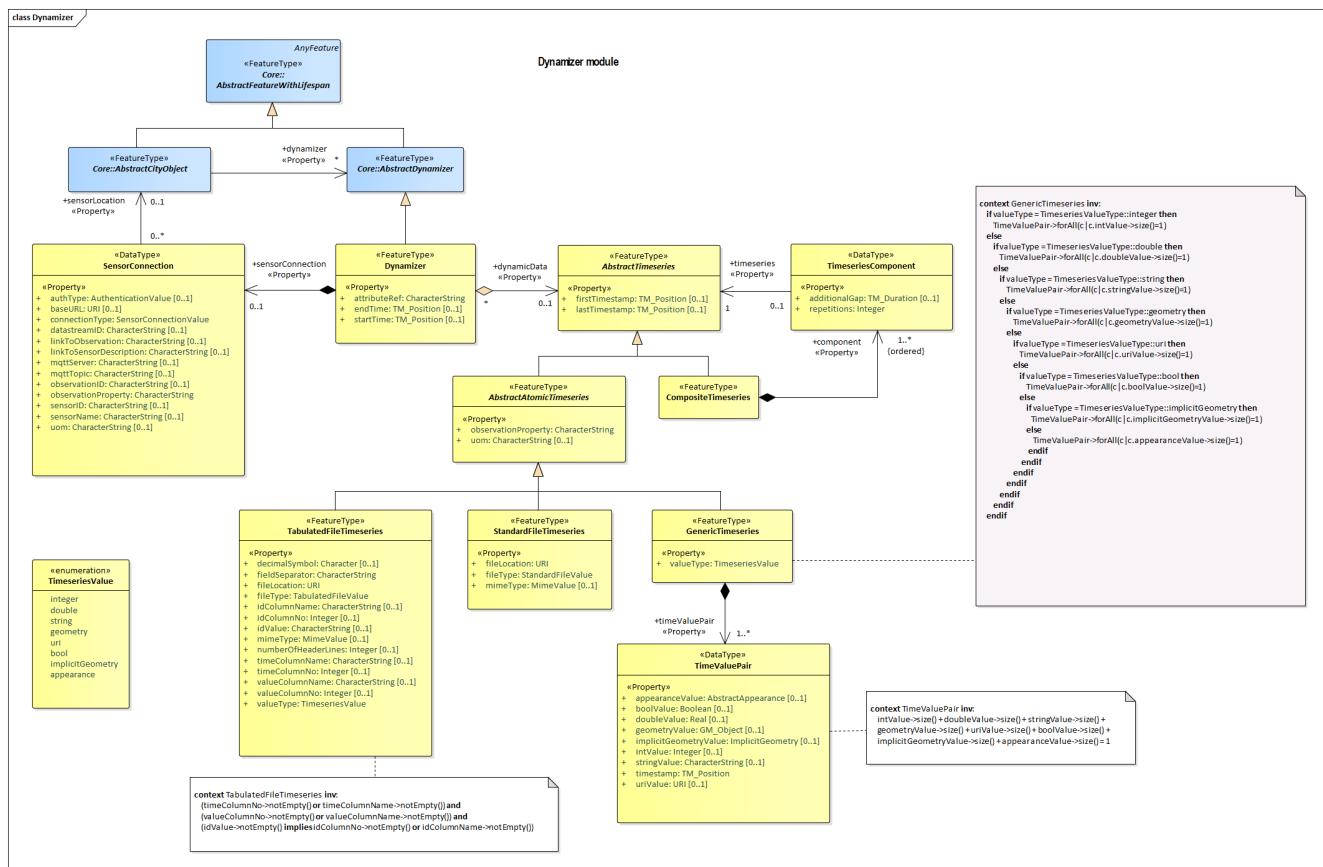


Figure 28. UML diagram of the Dynamizer Model.

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

Yellow	indicates
Blue	indicates
Pink	indicates

## 11.15.1. Dynamizer Package

### Package Dynamizer

Description:

Stereotype: «ApplicationSchema»

Parent Package: CityGML

## 11.15.2. Class AbstractAtomicTimeseries

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

### Class AbstractAtomicTimeseries

Definition:

Subclass Of: [AbstractTimeseries](#)

Stereotype: «FeatureType»

#### Roles

#### Attributes

Attribute Name: observationProperty

Value Type: CharacterString

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: uom

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

### 11.15.3. Class AbstractTimeseries

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 <a href="#">Dynamizer Data Dictionary</a> .

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

### Class AbstractTimeseries

Definition:

Subclass Of: <-- section,-->

Stereotype: «FeatureType»

### Roles

### 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»

## 11.15.4. Class AuthenticationValue

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 <a href="#">Dynamizer Data Dictionary</a> .

### Class AuthenticationValue

Definition:

Subclass Of: <-- section,-->

Stereotype: «CodeList»

### Roles

### Attributes

## 11.15.5. Class CompositeTimeseries

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

### Class CompositeTimeseries

Definition:

Subclass Of: [AbstractTimeseries](#)

Stereotype: «FeatureType»

#### Roles

Role Name: component

Cardinality: 1..\*

Target Class: [TimeseriesComponent](#)

Definition:

#### Attributes

## 11.15.6. Class Dynamizer

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

## Class Dynamizer

Definition:

Subclass Of: [AbstractDynamizer](#)

Stereotype: «FeatureType»

### Roles

Role Name: dynamicData

Cardinality: 0..1

Target Class: [AbstractTimeseries](#)

Definition:

Role Name: sensorConnection

Cardinality: 0..1

Target Class: [SensorConnection](#)

Definition:

### 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»

## 11.15.7. Class GenericTimeseries

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

### Class GenericTimeseries

Definition:

Subclass Of: [AbstractAtomicTimeseries](#)

Stereotype: «FeatureType»

#### Roles

Role Name: timeValuePair

Cardinality: 1..\*

Target Class: [TimeValuePair](#)

Definition:

#### Attributes

Attribute Name: valueType

Value Type: TimeseriesValue

Definition:

Multiplicity:

Stereotype: «Property»

### 11.15.8. Class SensorConnectionValue

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 <a href="#">Dynamizer Data Dictionary</a> .

### Class SensorConnectionValue

Definition:
Subclass Of: <-- section,>>
Stereotype: «CodeList»
<b>Roles</b>
<b>Attributes</b>

## 11.15.9. Class StandardFileTimeseries

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

<b>Class StandardFileTimeseries</b>
Definition:
Subclass Of: <a href="#">AbstractAtomicTimeseries</a>
Stereotype: «FeatureType»
<b>Roles</b>
<b>Attributes</b>
Attribute Name: fileLocation
Value Type: URI
Definition:
Multiplicity:
Stereotype: «Property»

Attribute Name:	fileType
Value Type:	StandardFileValue
Definition:	
Multiplicity:	
Stereotype:	«Property»

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

### 11.15.10. Class StandardFileValue

Requirement 293	/req/Dynamizer/StandardFileValue
A	Any use of the StandardFileValue type in an Implementation Specification SHALL be restricted to the valid values specified in the StandardFileValue UML class as documented in the StandardFileValue section of the <a href="#">Dynamizer Data Dictionary</a> .

#### Class StandardFileValue

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

#### Roles

#### Attributes

### 11.15.11. Class TabulatedFileTimeseries

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

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 <a href="#">Dynamizer Data Dictionary</a> .
---	---

## Class TabulatedFileTimeseries

Definition:

Subclass Of: [AbstractAtomicTimeseries](#)

Stereotype: «FeatureType»

### Roles

### Attributes

Attribute Name: decimalSymbol

Value Type: Character

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: fieldSeparator

Value Type: CharacterString

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: fileLocation

Value Type: URI

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: fileType

Value Type: TabulatedFileValue

Definition:

Multiplicity:

Stereotype: «Property»

Attribute Name: idColumnName

Value Type: CharacterString

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

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:	MimeType
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»

## 11.15.12. Class TabulatedFileValue

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 <a href="#">Dynamizer Data Dictionary</a> .
---	--

### Class TabulatedFileValue

Definition:

Subclass Of: <-- section,>>

Stereotype: «CodeList»

#### Roles

#### Attributes

### 11.15.13. Class SensorConnection

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

### Class SensorConnection

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

Role Name: sensorLocation

Cardinality: 0..1

Target Class: [AbstractCityObject](#)

Definition:

<b>Attributes</b>	
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»

#### 11.15.14. Class TimeseriesComponent

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

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 <a href="#">Dynamizer Data Dictionary</a> .
---	---

### Class TimeseriesComponent

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

#### Roles

Role Name: timeseries

Cardinality: 1

Target Class: [AbstractTimeseries](#)

Definition:

#### Attributes

Attribute Name: additionalGap

Value Type: TM\_Duration

Definition:

Multiplicity: [0..1]

Stereotype: «Property»

Attribute Name: repetitions

Value Type: Integer

Definition:

Multiplicity:

Stereotype: «Property»

### 11.15.15. Class TimeseriesValue

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 <a href="#">TimeseriesValue section of the Dynamizer Data Dictionary</a> .

### Class TimeseriesValue

Definition:

Subclass Of: <-- section,>>

Stereotype:

#### Roles

#### 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:

## 11.15.16. Class TimeValuePair

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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .
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 <a href="#">Dynamizer Data Dictionary</a> .

## Class TimeValuePair

Definition:

Subclass Of: <-- section,>>

Stereotype: «DataType»

### Roles

### 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»

## 11.15.17. 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 12. 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

<b>Abstract Test 1</b>	/ats/Appearance/AbstractSurfaceData
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-AbstractSurfaceData</a>
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.

<b>Abstract Test 2</b>	/ats/Appearance/AbstractTexture
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-AbstractTexture</a>
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.

<b>Abstract Test 3</b>	<a href="#"><b>/ats/Appearance/AbstractTextureParameterization</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Appearance/rc-AbstractTextureParameterization</b></a>
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.

<b>Abstract Test 4</b>	<a href="#"><b>/ats/Appearance/Appearance</b></a>
------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-Appearance</a>
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.

<b>Abstract Test 5</b>	<b>/ats/Appearance/Color</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-Color</a>
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.

<b>Abstract Test 6</b>	<a href="#"><b>/ats/Appearance/ColorPlusOpacity</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Appearance/rc-ColorPlusOpacity</b></a>
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.

<b>Abstract Test 7</b>	<a href="#"><b>/ats/Appearance/GeoreferencedTexture</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Appearance/rc-GeoreferencedTexture</a>
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.

<b>Abstract Test 8</b>	<a href="#">/ats/Appearance/ParameterizedTexture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-ParameterizedTexture</a>
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.

<b>Abstract Test 9</b>	<a href="#"><b>/ats/Appearance/TexCoordGen</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Appearance/rc-TexCoordGen</b></a>
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.

<b>Abstract Test 10</b>	<a href="#"><b>/ats/Appearance/TexCoordList</b></a>
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-TexCoordList</a>
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.

<b>Abstract Test 11</b>	<a href="#">/ats/Appearance/TextureAssociation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-TextureAssociation</a>
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.

<b>Abstract Test 12</b>	<b>/ats/Appearance/TextureType</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-TextureType</a>
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.

<b>Abstract Test 13</b>	<b>/ats/Appearance/WrapMode</b>
-------------------------	---------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-WrapMode</a>
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.

<b>Abstract Test 14</b>	<a href="#">/ats/Appearance/X3DMaterial</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Appearance/rc-X3DMaterial</a>
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

<b>Abstract Test 15</b>	<a href="#">/ats/Bridge/AbstractBridge</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-AbstractBridge</a>
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.

<b>Abstract Test 16</b>	<a href="#">/ats/Bridge/Bridge</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-Bridge</a>

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.

<b>Abstract Test 17</b>	<b>/ats/Bridge/BridgeClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeClassValue</a>
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

<b>Abstract Test 18</b>	<b>/ats/Bridge/BridgeConstructiveElement</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElement</a>
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.

<b>Abstract Test 19</b>	<a href="#">/ats/Bridge/BridgeConstructiveElementClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElementClassValue</a>
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

<b>Abstract Test 20</b>	<a href="#">/ats/Bridge/BridgeConstructiveElementFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElementFunctionValue</a>
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
---	---

<b>Abstract Test 21</b>	<a href="#">/ats/Bridge/BridgeConstructiveElementUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeConstructiveElementUsageValue</a>
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

<b>Abstract Test 22</b>	<a href="#">/ats/Bridge/BridgeFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFunctionValue</a>
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

<b>Abstract Test 23</b>	<a href="#">/ats/Bridge/BridgeFurniture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurniture</a>
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.

<b>Abstract Test 24</b>	<b>/ats/Bridge/BridgeFurnitureClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurnitureClassValue</a>
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

<b>Abstract Test 25</b>	<b>/ats/Bridge/BridgeFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurnitureFunctionValue</a>
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
---	---

<b>Abstract Test 26</b>	<a href="#">/ats/Bridge/BridgeFurnitureUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeFurnitureUsageValue</a>
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

<b>Abstract Test 27</b>	<a href="#">/ats/Bridge/BridgeInstallation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallation</a>
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.
---	---

<b>Abstract Test 28</b>	<b>/ats/Bridge/BridgeInstallationClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallationClassValue</a>
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

<b>Abstract Test 29</b>	<b>/ats/Bridge/BridgeInstallationFundntionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallationFundntionValue</a>
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

<b>Abstract Test 30</b>	<b>/ats/Bridge/BridgeInstallationUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeInstallationUsageValue</a>
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

<b>Abstract Test 31</b>	/ats/Bridge/BridgePart
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgePart</a>
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.

<b>Abstract Test 32</b>	/ats/Bridge/BridgeRoom
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoom</a>
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.

<b>Abstract Test 33</b>	<b>/ats/Bridge/BridgeRoomClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoomClassValue</a>
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

<b>Abstract Test 34</b>	<b>/ats/Bridge/BridgeRoomFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoomFunctionValue</a>
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

<b>Abstract Test 35</b>	<a href="#">/ats/Bridge/BridgeRoomUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeRoomUsageValue</a>
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

<b>Abstract Test 36</b>	<a href="#">/ats/Bridge/BridgeUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Bridge/rc-BridgeUsageValue</a>
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

<b>Abstract Test 37</b>	<a href="#">/ats/Building/AbstractBuilding</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-AbstractBuilding</a>
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.

<b>Abstract Test 38</b>	<a href="#">/ats/Building/AbstractBuildingSubdivision</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-AbstractBuildingSubdivision</a>
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.

<b>Abstract Test 39</b>	<a href="#">/ats/Building/Building</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Building/rc-Building</a>
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.

<b>Abstract Test 40</b>	<a href="#">/ats/Building/BuildingClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingClassValue</a>
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

<b>Abstract Test 41</b>	<a href="#">/ats/Building/BuildingConstructiveElement</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElement</a>

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.

<b>Abstract Test 42</b>	<a href="#">/ats/Building/BuildingConstructiveElementClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElementClassValue</a>
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

<b>Abstract Test 43</b>	<a href="#">/ats/Building/BuildingConstructiveElementFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingConstructiveElementFunctionValue</a>

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

<b>Abstract Test 44</b>	<a href="#"><b>/ats/Building/BuildingConstructiveElementUsageValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Building/rc-BuildingConstructiveElementUsageValue</i></a>
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

<b>Abstract Test 45</b>	<a href="#"><b>/ats/Building/BuildingFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Building/rc-BuildingFunctionValue</i></a>
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

<b>Abstract Test 46</b>	<a href="#"><b>/ats/Building/BuildingFurniture</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Building/rc-BuildingFurniture</i></a>
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.

<b>Abstract Test 47</b>	<b>/ats/Building/BuildingFurnitureClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Building/rc-BuildingFurnitureClassValue</i></a>
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

<b>Abstract Test 48</b>	<b>/ats/Building/BuildingFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Building/rc-BuildingFurnitureFunctionValue</i></a>
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
---	---

<b>Abstract Test 49</b>	<a href="#">/ats/Building/BuildingFurnitureUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingFurnitureUsageValue</a>
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

<b>Abstract Test 50</b>	<a href="#">/ats/Building/BuildingInstallation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallation</a>
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.
---	---

<b>Abstract Test 51</b>	<b>/ats/Building/BuildingInstallationClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallationClassValue</a>
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

<b>Abstract Test 52</b>	<b>/ats/Building/BuildingInstallationFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallationFunctionValue</a>
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

<b>Abstract Test 53</b>	<b>/ats/Building/BuildingInstallationUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingInstallationUsageValue</a>
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
---	---

<b>Abstract Test 54</b>	<a href="#">/ats/Building/BuildingPart</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingPart</a>
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.

<b>Abstract Test 55</b>	<a href="#">/ats/Building/BuildingRoom</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingRoom</a>
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.

<b>Abstract Test 56</b>	<a href="#"><b>/ats/Building/BuildingRoomClassValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Building/rc-BuildingRoomClassValue</b></a>
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

<b>Abstract Test 57</b>	<a href="#"><b>/ats/Building/BuildingRoomFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Building/rc-BuildingRoomFunctionValue</b></a>
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

<b>Abstract Test 58</b>	<a href="#">/ats/Building/BuildingRoomUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingRoomUsageValue</a>
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

<b>Abstract Test 59</b>	<a href="#">/ats/Building/BuildingSubdivisionClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingSubdivisionClassValue</a>
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

<b>Abstract Test 60</b>	<a href="#">/ats/Building/BuildingSubdivisionFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingSubdivisionFunctionValue</a>
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

<b>Abstract Test 61</b>	<a href="#">/ats/Building/BuildingSubdivisionUsageValue</a>
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingSubdivisionUsageValue</a>
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

<b>Abstract Test 62</b>	<a href="#">/ats/Building/BuildingUnit</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-BuildingUnit</a>
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.

<b>Abstract Test 63</b>	<a href="#">/ats/Building/BuildingUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Building/rc-BuildingUsageValue</a>
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

<b>Abstract Test 64</b>	<b><a href="#">/ats/Building/RoofTypeValue</a></b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-RoofTypeValue</a>
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

<b>Abstract Test 65</b>	<b><a href="#">/ats/Building/RoomElevationReferenceValue</a></b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-RoomElevationReferenceValue</a>
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

<b>Abstract Test 66</b>	<b><a href="#">/ats/Building/RoomHeight</a></b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-RoomHeight</a>

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.

<b>Abstract Test 67</b>	<a href="#">/ats/Building/Storey</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Building/rc-Storey</a>
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

<b>Abstract Test 68</b>	<a href="#">/ats/CityFurniture/CityFurniture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurniture</a>
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.

<b>Abstract Test 69</b>	<a href="#">/ats/CityFurniture/CityFurnitureClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurnitureClassValue</a>
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
---	--

<b>Abstract Test 70</b>	<b>/ats/CityFurniture/CityFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurnitureFunctionValue</a>
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

<b>Abstract Test 71</b>	<b>/ats/CityFurniture/CityFurnitureUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityFurniture/rc-CityFurnitureUsageValue</a>
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

<b>Abstract Test 72</b>	<b>/ats/CityObjectGroup/CityObjectGroup</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroup</a>
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.

<b>Abstract Test 73</b>	<b>/ats/CityObjectGroup/CityObjectGroupClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroupClassValue</a>
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

<b>Abstract Test 74</b>	<b>/ats/CityObjectGroup/CityObjectGroupFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroupFunctionValue</a>
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
---	---

<b>Abstract Test 75</b>	<a href="#">/ats/CityObjectGroup/CityObjectGroupUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-CityObjectGroupUsageValue</a>
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

<b>Abstract Test 76</b>	<a href="#">/ats/CityObjectGroup/Role</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/CityObjectGroup/rc-Role</a>
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	<a href="#">/ats/Core/AbstractAppearance</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractAppearance</a>
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	<a href="#">/ats/Core/AbstractCityObject</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractCityObject</a>
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.

<b>Abstract Test 79</b>	<a href="#">/ats/Core/AbstractDynamizer</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractDynamizer</a>
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.

<b>Abstract Test 80</b>	<a href="#">/ats/Core/AbstractFeatureWithLifespan</a>
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractFeatureWithLifespan</a>
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.

<b>Abstract Test 81</b>	<a href="#">/ats/Core/AbstractGenericAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractGenericAttribute</a>
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.

<b>Abstract Test 82</b>	<a href="#">/ats/Core/AbstractLogicalSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractLogicalSpace</a>
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.

<b>Abstract Test 83</b>	<a href="#">/ats/Core/AbstractOccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractOccupiedSpace</a>

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.

<b>Abstract Test 84</b>	<a href="#">/ats/Core/AbstractPhysicalSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractPhysicalSpace</a>
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.

<b>Abstract Test 85</b>	/ats/Core/AbstractPointCloud
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractPointCloud</a>
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.

<b>Abstract Test 86</b>	/ats/Core/AbstractSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractSpace</a>
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.
---	--

<b>Abstract Test 87</b>	<a href="#">/ats/Core/AbstractSpaceBoundary</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractSpaceBoundary</a>
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.

<b>Abstract Test 88</b>	<a href="#">/ats/Core/AbstractThematicSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractThematicSurface</a>
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.

<b>Abstract Test 89</b>	<a href="#">/ats/Core/AbstractUnoccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractUnoccupiedSpace</a>
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.

<b>Abstract Test 90</b>	<a href="#">/ats/Core/AbstractVersion</a>
-------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractVersion</a>
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.

<b>Abstract Test 91</b>	<a href="#">/ats/Core/AbstractVersionTransition</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-AbstractVersionTransition</a>
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.
---	--

<b>Abstract Test 92</b>	/ats/Core/Address
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-Address</a>
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.

<b>Abstract Test 93</b>	/ats/Core/CityModel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-CityModel</a>
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.

<b>Abstract Test 94</b>	<a href="#">/ats/Core/CityObjectRelation</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-CityObjectRelation</a>
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.
---	--

<b>Abstract Test 95</b>	<b>/ats/Core/ClosureSurface</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-ClosureSurface</a>
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.

<b>Abstract Test 96</b>	<b>/ats/Core/DoubleBetween0and1</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-DoubleBetween0and1</a>
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.

<b>Abstract Test 97</b>	<b>/ats/Core/DoubleBetween0and1List</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Core/rc-DoubleBetween0and1List</b></a>
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.
---	--

<b>Abstract Test 98</b>	<b>/ats/Core/DoubleList</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-DoubleList</a>
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.

<b>Abstract Test 99</b>	<b>/ats/Core/ExternalReference</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-ExternalReference</a>
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.

<b>Abstract Test 100</b>	<a href="#">/ats/Core/ImplicitGeometry</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-ImplicitGeometry</a>
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.
---	---

<b>Abstract Test 101</b>	<a href="#">/ats/Core/IntegerBetween0and3</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-IntegerBetween0and3</a>
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.

<b>Abstract Test 102</b>	<a href="#">/ats/Core/IntervalValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-IntervalValue</a>
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
---	--

<b>Abstract Test 103</b>	<a href="#">/ats/Core/MimeValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-MimeValue</a>
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

<b>Abstract Test 104</b>	<a href="#">/ats/Core/Occupancy</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-Occupancy</a>
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.
---	---

<b>Abstract Test 105</b>	<a href="#"><b>/ats/Core/OccupantTypeValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Core/rc-OccupantTypeValue</i></a>
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

<b>Abstract Test 106</b>	<a href="#"><b>/ats/Core/OtherRelationTypeValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Core/rc-OtherRelationTypeValue</i></a>
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

<b>Abstract Test 107</b>	<a href="#"><b>/ats/Core/QualifiedArea</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#"><i>/req/Core/rc-QualifiedArea</i></a>
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.

<b>Abstract Test 108</b>	<a href="#">/ats/Core/QualifiedAreaValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedAreaValue</a>
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

<b>Abstract Test 109</b>	<a href="#">/ats/Core/QualifiedVolume</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedVolume</a>
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.

<b>Abstract Test 110</b>	<b>/ats/Core/QualifiedVolumeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-QualifiedVolumeValue</a>
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

<b>Abstract Test 111</b>	<b>/ats/Core/RelationTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-RelationTypeValue</a>
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

<b>Abstract Test 112</b>	<a href="#">/ats/Core/RelativeToTerrain</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-RelativeToTerrain</a>
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

<b>Abstract Test 113</b>	<a href="#">/ats/Core/RelativeToWater</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-RelativeToWater</a>
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

<b>Abstract Test 114</b>	<a href="#">/ats/Core/SpaceType</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-SpaceType</a>
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

<b>Abstract Test 115</b>	<a href="#">/ats/Core/TemporalRelationTypeValue</a>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TemporalRelationTypeValue</a>
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

<b>Abstract Test 116</b>	<b>/ats/Core/TopologicalRelationTypeValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TopologicalRelationTypeValue</a>
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

<b>Abstract Test 117</b>	<b>/ats/Core/TransformationMatrix2x2</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TransformationMatrix2x2</a>
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.

<b>Abstract Test 118</b>	<b>/ats/Core/TransformationMatrix3x4</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TransformationMatrix3x4</a>
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.

<b>Abstract Test 119</b>	<b>/ats/Core/TransformationMatrix4x4</b>
--------------------------	--

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-TransformationMatrix4x4</a>
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.

<b>Abstract Test 120</b>	<a href="#">/ats/Core/XALAddressDetails</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Core/rc-XALAddressDetails</a>
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

<b>Abstract Test 121</b>	<a href="#">/ats/Dynamizer/AbstractAtomicTimeSeries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-AbstractAtomicTimeSeries</a>
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.

<b>Abstract Test 122</b>	<a href="#">/ats/Dynamizer/AbstractTimeSeries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-AbstractTimeSeries</a>
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.

<b>Abstract Test 123</b>	<a href="#">/ats/Dynamizer/AuthenticationValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-AuthenticationValue</a>
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

<b>Abstract Test 124</b>	<a href="#">/ats/Dynamizer/CompositeTimeseries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Dynamizer/rc-CompositeTimeseries</a>
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.

<b>Abstract Test 125</b>	<a href="#">/ats/Dynamizer/Dynamizer</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-Dynamizer</a>
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.

<b>Abstract Test 126</b>	<a href="#">/ats/Dynamizer/GenericTimeseries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-GenericTimeseries</a>
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.

<b>Abstract Test 127</b>	<a href="#">/ats/Dynamizer/SensorConnection</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Dynamizer/rc-SensorConnection</a>
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.

<b>Abstract Test 128</b>	<a href="#">/ats/Dynamizer/SensorConnectionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-SensorConnectionValue</a>
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

<b>Abstract Test 129</b>	<a href="#">/ats/Dynamizer/StandardFileTimeseries</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-StandardFileTimeseries</a>

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.

<b>Abstract Test 130</b>	<b>/ats/Dynamizer/StandardFieldValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-StandardFieldValue</a>
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

<b>Abstract Test 131</b>	<b>/ats/Dynamizer/TabulatedFileTimeseries</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TabulatedFileTimeseries</a>
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.

<b>Abstract Test 132</b>	<a href="#">/ats/Dynamizer/TabulatedFileValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TabulatedFileValue</a>
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

<b>Abstract Test 133</b>	<a href="#">/ats/Dynamizer/TimeseriesComponent</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeseriesComponent</a>
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.

<b>Abstract Test 134</b>	<a href="#">/ats/Dynamizer/TimeseriesValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeseriesValue</a>
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

<b>Abstract Test 135</b>	<a href="#">/ats/Dynamizer/TimeValuePair</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Dynamizer/rc-TimeValuePair</a>
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

<b>Abstract Test 136</b>	<a href="#">/ats/Generics/DateAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-DateAttribute</a>
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.

<b>Abstract Test 137</b>	<a href="#">/ats/Generics/DoubleAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-DoubleAttribute</a>
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.

<b>Abstract Test 138</b>	<a href="#">/ats/Generics/GenericAttributeSet</a>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericAttributeSet</a>
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.

<b>Abstract Test 139</b>	<a href="#">/ats/Generics/GenericLogicalSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpace</a>
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.

<b>Abstract Test 140</b>	<a href="#">/ats/Generics/GenericLogicalSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpaceClassValue</a>
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

<b>Abstract Test 141</b>	<a href="#">/ats/Generics/GenericLogicalSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpaceFunctionValue</a>
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

<b>Abstract Test 142</b>	<a href="#">/ats/Generics/GenericLogicalSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Generics/rc-GenericLogicalSpaceUsageValue</a>
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

<b>Abstract Test 143</b>	<a href="#">/ats/Generics/GenericOccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericOccupiedSpace</a>
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.

<b>Abstract Test 144</b>	<a href="#">/ats/Generics/GenericOccupiedSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericOccupiedSpaceClassValue</a>

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

<b>Abstract Test 145</b>	<a href="#"><b>/ats/Generics/GenericOccupiedSpaceFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Generics/rc-GenericOccupiedSpaceFunctionValue</b></a>
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

<b>Abstract Test 146</b>	<a href="#"><b>/ats/Generics/GenericOccupiedSpaceUsageValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Generics/rc-GenericOccupiedSpaceUsageValue</b></a>
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

<b>Abstract Test 147</b>	<a href="#"><b>/ats/Generics/GenericThematicSurface</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Generics/rc-GenericThematicSurface</b></a>
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.

<b>Abstract Test 148</b>	<a href="#">/ats/Generics/GenericThematicSurfaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurfaceClassValue</a>
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

<b>Abstract Test 149</b>	<a href="#">/ats/Generics/GenericThematicSurfaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurfaceFunctionValue</a>
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
---	--

<b>Abstract Test 150</b>	<a href="#">/ats/Generics/GenericThematicSurfaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericThematicSurfaceUsageValue</a>
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

<b>Abstract Test 151</b>	<a href="#">/ats/Generics/GenericUnoccupiedSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpace</a>
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	<a href="#">/ats/Generics/GenericUnoccupiedSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpaceClassValue</a>
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	<a href="#">/ats/Generics/GenericUnoccupiedSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpaceFunctionValue</a>
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	<a href="#">/ats/Generics/GenericUnoccupiedSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-GenericUnoccupiedSpaceUsageValue</a>
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
---	---

<b>Abstract Test 155</b>	<a href="#">/ats/Generics/IntAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-IntAttribute</a>
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.

<b>Abstract Test 156</b>	<a href="#">/ats/Generics/MeasureAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-MeasureAttribute</a>
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.

<b>Abstract Test 157</b>	<a href="#">/ats/Generics/StringAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-StringAttribute</a>
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.
---	---

<b>Abstract Test 158</b>	<a href="#">/ats/Generics/UriAttribute</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Generics/rc-UriAttribute</a>
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

<b>Abstract Test 159</b>	<a href="#">/ats/LandUse/LandUse</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUse</a>
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.

<b>Abstract Test 160</b>	<b>/ats/LandUse/LandUseClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUseClassValue</a>
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

<b>Abstract Test 161</b>	<b>/ats/LandUse/LandUseFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUseFunctionValue</a>
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
---	---

<b>Abstract Test 162</b>	<a href="#">/ats/LandUse/LandUseUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/LandUse/rc-LandUseUsageValue</a>
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

<b>Abstract Test 163</b>	<a href="#">/ats/PointCloud/PointCloud</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/PointCloud/rc-PointCloud</a>
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

<b>Abstract Test 164</b>	<a href="#">/ats/Relief/AbstractReliefComponent</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-AbstractReliefComponent</a>
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.

<b>Abstract Test 165</b>	<a href="#">/ats/Relief/BreaklineRelief</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-BreaklineRelief</a>
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.

<b>Abstract Test 166</b>	<b>/ats/Relief/MassPointRelief</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-MassPointRelief</a>
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.

<b>Abstract Test 167</b>	<a href="#">/ats/Relief/RasterRelief</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-RasterRelief</a>
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.

<b>Abstract Test 168</b>	<a href="#">/ats/Relief/ReliefFeature</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-ReliefFeature</a>
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.

<b>Abstract Test 169</b>	<a href="#">/ats/Relief/TINRelief</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Relief/rc-TINRelief</a>
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

<b>Abstract Test 170</b>	/ats/Transportation/AbstractTransportationSpace
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AbstractTransportationSpace</a>
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.

<b>Abstract Test 171</b>	/ats/Transportation/AuxiliaryTrafficArea
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficArea</a>
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.

<b>Abstract Test 172</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficAreaClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficAreaClassValue</a>
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

<b>Abstract Test 173</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficAreaFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficAreaFunctionValue</a>
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

<b>Abstract Test 174</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficAreaUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficAreaUsageValue</a>
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

<b>Abstract Test 175</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpace</a>
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.

<b>Abstract Test 176</b>	<a href="#">/ats/Transportation/AuxiliaryTrafficSpaceClassValue</a>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpaceClassValue</a>
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

<b>Abstract Test 177</b>	<b>/ats/Transportation/AuxiliaryTrafficSpaceFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpaceFunctionValue</a>
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

<b>Abstract Test 178</b>	<b>/ats/Transportation/AuxiliaryTrafficSpaceUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-AuxiliaryTrafficSpaceUsageValue</a>
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

<b>Abstract Test 179</b>	<b>/ats/Transportation/ClearanceSpace</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Transportation/rc-ClearanceSpace</a>
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.

<b>Abstract Test 180</b>	<a href="#">/ats/Transportation/ClearanceSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-ClearanceSpaceClassValue</a>
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

<b>Abstract Test 181</b>	<a href="#">/ats/Transportation/GranularityValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-GranularityValue</a>

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

<b>Abstract Test 182</b>	/ats/Transportation/Hole
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Hole</a>
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.

<b>Abstract Test 183</b>	/ats/Transportation/HoleClassName
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-HoleClassName</a>
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
---	---

<b>Abstract Test 184</b>	<a href="#">/ats/Transportation/HoleSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-HoleSurface</a>
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.

<b>Abstract Test 185</b>	<a href="#">/ats/Transportation/Intersection</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Intersection</a>
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.

<b>Abstract Test 186</b>	<a href="#">/ats/Transportation/IntersectionClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-IntersectionClassValue</a>
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

<b>Abstract Test 187</b>	<a href="#">/ats/Transportation/Marking</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Marking</a>
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.

<b>Abstract Test 188</b>	<a href="#">/ats/Transportation/MarkingClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-MarkingClassValue</a>
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

<b>Abstract Test 189</b>	<a href="#">/ats/Transportation/Railway</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Railway</a>
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.

<b>Abstract Test 190</b>	<a href="#">/ats/Transportation/RailwayClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RailwayClassValue</a>
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

<b>Abstract Test 191</b>	<a href="#">/ats/Transportation/RailwayFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RailwayFunctionValue</a>
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

<b>Abstract Test 192</b>	<a href="#">/ats/Transportation/RailwayUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RailwayUsageValue</a>
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

<b>Abstract Test 193</b>	<a href="#">/ats/Transportation/Road</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Road</a>
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.

<b>Abstract Test 194</b>	<a href="#">/ats/Transportation/RoadClassValue</a>
--------------------------	--

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RoadClassValue</a>
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

<b>Abstract Test 195</b>	<b>/ats/Transportation/RoadFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RoadFunctionValue</a>
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

<b>Abstract Test 196</b>	<b>/ats/Transportation/RoadUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-RoadUsageValue</a>
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

<b>Abstract Test 197</b>	<b>/ats/Transportation/Section</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Transportation/rc-Section</a>
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.

<b>Abstract Test 198</b>	<a href="#">/ats/Transportation/SectionClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SectionClassValue</a>
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

<b>Abstract Test 199</b>	<a href="#">/ats/Transportation/Square</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Square</a>

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.

<b>Abstract Test 200</b>	<a href="#">/ats/Transportation/SquareClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SquareClassValue</a>
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

<b>Abstract Test 201</b>	<a href="#">/ats/Transportation/SquareFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SquareFunctionValue</a>
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
---	--

<b>Abstract Test 202</b>	<a href="#">/ats/Transportation/SquareUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SquareUsageValue</a>
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

<b>Abstract Test 203</b>	<a href="#">/ats/Transportation/SurfaceMaterialValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-SurfaceMaterialValue</a>
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

<b>Abstract Test 204</b>	<a href="#">/ats/Transportation/Track</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Track</a>
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.

<b>Abstract Test 205</b>	<a href="#"><b>/ats/Transportation/TrackClassValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TrackClassValue</b></a>
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

<b>Abstract Test 206</b>	<a href="#"><b>/ats/Transportation/TrackFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TrackFunctionValue</b></a>
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

<b>Abstract Test 207</b>	<a href="#">/ats/Transportation/TrackUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrackUsageValue</a>
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

<b>Abstract Test 208</b>	<a href="#">/ats/Transportation/TrafficArea</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficArea</a>
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.

<b>Abstract Test 209</b>	<a href="#">/ats/Transportation/TrafficAreaClassValue</a>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficAreaClassValue</a>
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

<b>Abstract Test 210</b>	<b>/ats/Transportation/TrafficAreaFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficAreaFunctionValue</a>
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

<b>Abstract Test 211</b>	<b>/ats/Transportation/TrafficAreaUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficAreaUsageValue</a>
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

<b>Abstract Test 212</b>	<b>/ats/Transportation/TrafficDirectionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Transportation/rc-TrafficDirectionValue</a>
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

<b>Abstract Test 213</b>	<a href="#">/ats/Transportation/TrafficSpace</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficSpace</a>
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.

<b>Abstract Test 214</b>	<a href="#">/ats/Transportation/TrafficSpaceClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TrafficSpaceClassValue</a>

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

<b>Abstract Test 215</b>	<a href="#"><b>/ats/Transportation/TrafficSpaceFunctionValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TrafficSpaceFunctionValue</b></a>
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

<b>Abstract Test 216</b>	<a href="#"><b>/ats/Transportation/TrafficSpaceUsageValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TrafficSpaceUsageValue</b></a>
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

<b>Abstract Test 217</b>	<a href="#"><b>/ats/Transportation/TransportationSpaceClassValue</b></a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#"><b>/req/Transportation/rc-TransportationSpaceClassValue</b></a>
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
---	--

<b>Abstract Test 218</b>	<a href="#">/ats/Transportation/TransportationSpaceFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TransportationSpaceFunctionValue</a>
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

<b>Abstract Test 219</b>	<a href="#">/ats/Transportation/TransportationSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-TransportationSpaceUsageValue</a>
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

<b>Abstract Test 220</b>	<a href="#">/ats/Transportation/Waterway</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-Waterway</a>
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.

<b>Abstract Test 221</b>	<a href="#">/ats/Transportation/WaterwayClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-WaterwayClassValue</a>
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

<b>Abstract Test 222</b>	<a href="#">/ats/Transportation/WaterwayFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-WaterwayFunctionValue</a>
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

<b>Abstract Test 223</b>	/ats/Transportation/WaterwayUsageValue
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Transportation/rc-WaterwayUsageValue</a>
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

<b>Abstract Test 224</b>	/ats/Tunnel/AbstractTunnel
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-AbstractTunnel</a>
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.

<b>Abstract Test 225</b>	/ats/Tunnel/HollowSpace
--------------------------	-------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-HollowSpace</a>
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.

<b>Abstract Test 226</b>	<b>/ats/Tunnel/HollowSpaceClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-HollowSpaceClassValue</a>
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

<b>Abstract Test 227</b>	<b>/ats/Tunnel/HollowSpaceFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Tunnel/rc-HollowSpaceFunctionValue</a>
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

<b>Abstract Test 228</b>	<a href="#">/ats/Tunnel/HollowSpaceUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-HollowSpaceUsageValue</a>
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

<b>Abstract Test 229</b>	<a href="#">/ats/Tunnel/Tunnel</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-Tunnel</a>
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.

<b>Abstract Test 230</b>	<a href="#">/ats/Tunnel/TunnelClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelClassValue</a>
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

<b>Abstract Test 231</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElement</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElement</a>
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.

<b>Abstract Test 232</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElementClassValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElementClassValue</a>
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

<b>Abstract Test 233</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElementFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElementFunctionValue</a>
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

<b>Abstract Test 234</b>	<a href="#">/ats/Tunnel/TunnelConstructiveElementUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model

Requirement	<a href="#">/req/Tunnel/rc-TunnelConstructiveElementUsageValue</a>
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

<b>Abstract Test 235</b>	<a href="#">/ats/Tunnel/TunnelFunctionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFunctionValue</a>
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

<b>Abstract Test 236</b>	<a href="#">/ats/Tunnel/TunnelFurniture</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurniture</a>
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.

<b>Abstract Test 237</b>	<b>/ats/Tunnel/TunnelFurnitureClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurnitureClassValue</a>
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

<b>Abstract Test 238</b>	<b>/ats/Tunnel/TunnelFurnitureFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurnitureFunctionValue</a>
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

<b>Abstract Test 239</b>	<b>/ats/Tunnel/TunnelFurnitureUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelFurnitureUsageValue</a>

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

<b>Abstract Test 240</b>	<b>/ats/Tunnel/TunnelInstallation</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallation</a>
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.

<b>Abstract Test 241</b>	<b>/ats/Tunnel/TunnelInstallationClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallationClassValue</a>
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
---	---

<b>Abstract Test 242</b>	<b>/ats/Tunnel/TunnelInstallationFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallationFunctionValue</a>
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

<b>Abstract Test 243</b>	<b>/ats/Tunnel/TunnelInstallationUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelInstallationUsageValue</a>
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

<b>Abstract Test 244</b>	<b>/ats/Tunnel/TunnelPart</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelPart</a>
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.

<b>Abstract Test 245</b>	<b>/ats/Tunnel/TunnelUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Tunnel/rc-TunnelUsageValue</a>
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

<b>Abstract Test 246</b>	<b>/ats/Vegetation/AbstractVegetationObject</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-AbstractVegetationObject</a>
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.

<b>Abstract Test 247</b>	<a href="#">/ats/Vegetation/PlantCover</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCover</a>
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.

<b>Abstract Test 248</b>	<b>/ats/Vegetation/PlantCoverClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCoverClassValue</a>
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

<b>Abstract Test 249</b>	<b>/ats/Vegetation/PlantCoverFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCoverFunctionValue</a>
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

<b>Abstract Test 250</b>	<b>/ats/Vegetation/PlantCoverUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-PlantCoverUsageValue</a>
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

<b>Abstract Test 251</b>	<b>/ats/Vegetation/SolitaryVegetationObject</b>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObject</a>
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.

<b>Abstract Test 252</b>	<b>/ats/Vegetation/SolitaryVegetationObjectClassValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObjectClassValue</a>
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

<b>Abstract Test 253</b>	<b>/ats/Vegetation/SolitaryVegetationObjectFunctionValue</b>
--------------------------	--

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObjectFunctionValue</a>
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

<b>Abstract Test 254</b>	<a href="#">/ats/Vegetation/SolitaryVegetationObjectUsageValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SolitaryVegetationObjectUsageValue</a>
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

<b>Abstract Test 255</b>	<a href="#">/ats/Vegetation/SpeciesValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Vegetation/rc-SpeciesValue</a>
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

<b>Abstract Test 256</b>	<a href="#">/ats/Versioning/Transaction</a>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Basic Type Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-Transaction</a>
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.

<b>Abstract Test 257</b>	<a href="#">/ats/Versioning/TransactionValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-TransactionValue</a>
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

<b>Abstract Test 258</b>	<a href="#">/ats/Versioning/TransitionValue</a>
--------------------------	---

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-TransitionValue</a>
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

<b>Abstract Test 259</b>	<a href="#">/ats/Versioning/Version</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Versioning/rc-Version</a>
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.

<b>Abstract Test 260</b>	<a href="#">/ats/Versioning/VersionTransition</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Versioning/rc-VersionTransition</a>
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	<a href="#">/ats/Waterbody/AbstractWaterBoundarySurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Abstract Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-AbstractWaterBoundarySurface</a>
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.

<b>Abstract Test 262</b>	/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.

<b>Abstract Test 263</b>	/ats/Waterbody/WaterBodyClassValue
--------------------------	------------------------------------

Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterBodyClassValue</a>
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

<b>Abstract Test 264</b>	<b>/ats/Waterbody/WaterBodyFunctionValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterBodyFunctionValue</a>
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

<b>Abstract Test 265</b>	<b>/ats/Waterbody/WaterBodyUsageValue</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterBodyUsageValue</a>
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

<b>Abstract Test 266</b>	<b>/ats/Waterbody/WaterGroundSurface</b>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model

Requirement	<a href="#">/req/Waterbody/rc-WaterGroundSurface</a>
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.

<b>Abstract Test 267</b>	<a href="#">/ats/Waterbody/WaterLevelValue</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Codelists defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterLevelValue</a>
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

<b>Abstract Test 268</b>	<a href="#">/ats/Waterbody/WaterSurface</a>
Test Purpose	To validate that the Implementation Specification correctly implements the UML Classes defined in the Conceptual Model
Requirement	<a href="#">/req/Waterbody/rc-WaterSurface</a>

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 13. 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).