GEOSPATIAL AND BUILDING STANDARDS

Corso di: Big Data Architectures

Dott. Marco Fanfani

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

http://www.disit.dinfo.unifi.it/

Dipartimento di Ingegneria dell'Informazione

Università degli Studi di Firenze

CC BY SA

WHY 3D DATA MODELS MATTER FOR DIGITAL TWINS

A Digital Twin is much more than a 3D visualization.

It requires the integration of multiple heterogeneous data sources:

- Building Information Models (BIM)
- Geographic Information Systems (GIS)
- IoT sensors, PoI, real-time data, etc.

Each domain has been developed independently and adopts different:

- data models
- semantic representations
- geometric abstractions
- interoperability standards

How can these heterogeneous models be integrated into a single, scalable Digital Twin platform?

THE EVOLUTION OF 3D STANDARDS

Standard	Primary Goal	Main Strength	Main Limitation
IFC	Building design	Rich engineering semantics	Limited city-scale management
GIS	Spatial analysis	Geographical context	Limited building detail
CityGML	Semantic 3D cities	Urban-scale representation	Static and XML-based
3D Tiles / glTF	Web visualization	Efficient streaming and rendering	Very limited semantics

The goal is to have simultaneously:

- rich engineering semantics
- urban-scale geospatial context
- efficient web visualization
- real-time interoperability
- integration with IoT and knowledge graphs

This motivates the architecture adopted by **Snap4City**, where visualization, semantics and knowledge management are separated but tightly connected.

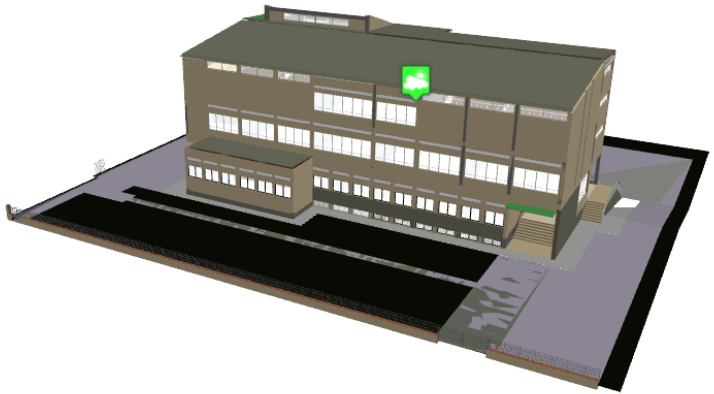
WHAT IS A BIM?



BIM is a **digital representation of a building** as an information system. Not only a 3D model, but a **structured database of building objects.**

Each element combines:

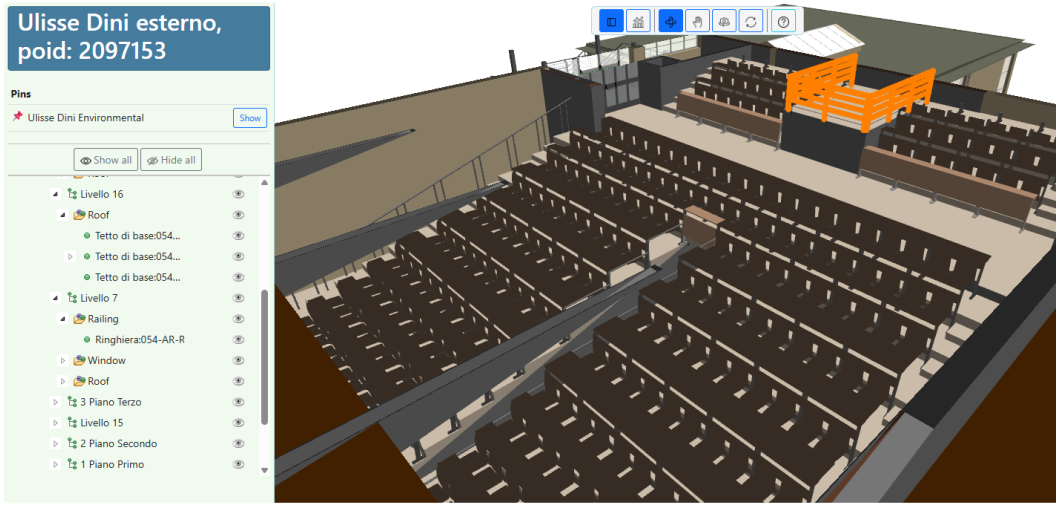
- geometry (shape, dimensions)
- semantics (what the object is)
- properties (materials, performance, cost, maintenance data)



WHAT IS A BIM?

Object-based modelling paradigm

- Buildings are modelled as **interacting objects, not drawings**
- Examples: walls, slabs, beams, doors, HVAC systems
- Each object has a unique identity and attributes



WHAT IS A BIM?

Lifecycle-oriented approach

- BIM supports the full asset lifecycle:

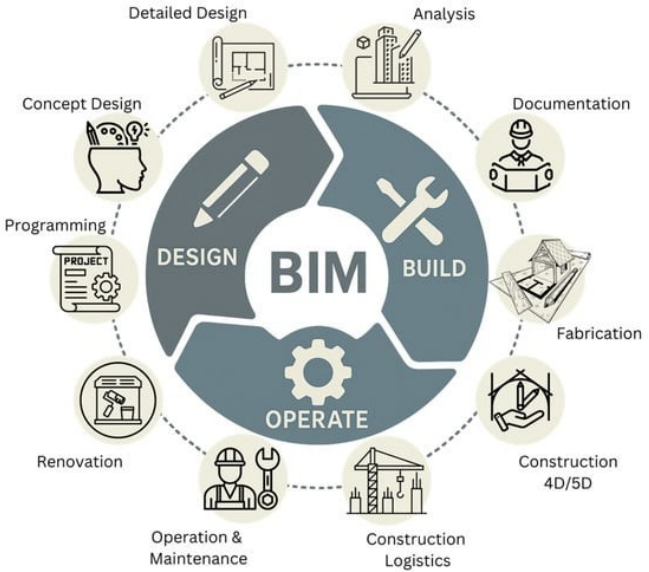
  design → construction → operation → renovation → demolition

- Information is **continuously enriched** over time

Interoperability challenge

- BIM tools are heterogeneous
- Data exchange requires standardized formats

  → **IFC (Industry Foundation Classes)**



Credit: Al Aamri, et al. Barriers and Opportunities for the Adoption of Building Information Modelling in the Design of Buildings: Case Study of Oman. *Sustainability* **2025**

THE IFC DATA MODEL

IFC is an open standard data schema for BIM interoperability

- Developed by buildingSMART
- Defines a **neutral, software-independent representation of building data**
- Enables exchange without losing semantic meaning

IFC is a semantic data model (not just file format)

- IFC is based on an **object-oriented schema (EXPRESS + STEP physical encoding)**
- Data is structured as:
  - objects (entities)
  - attributes
  - relationships

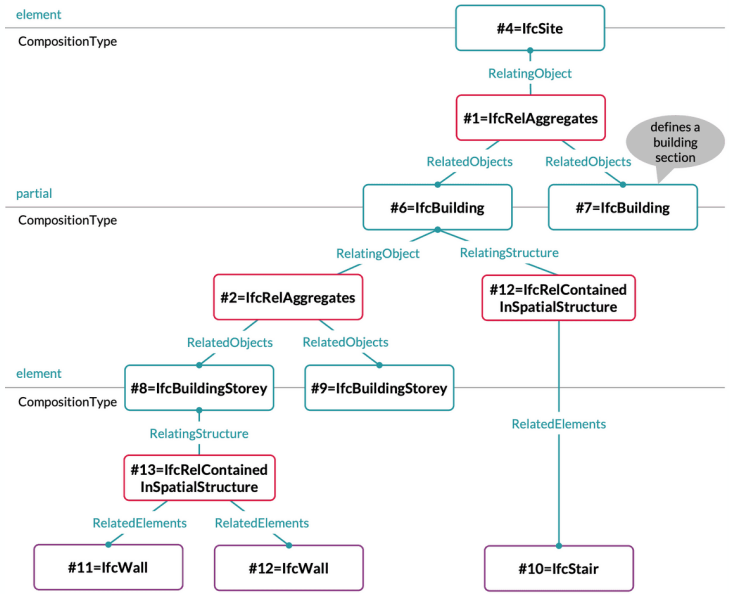
THE IFC DATA MODEL

Main components of IFC structure

- Spatial hierarchy:
  - IfcProject → IfcSite → IfcBuilding → IfcBuildingStorey
- Elements:
  - IfcWall, IfcDoor, IfcWindow, IfcSlab, etc.
- Relationships:
  - IfcRelAggregates (hierarchy)
  - IfcRelContainedInSpatialStructure (spatial containment)
  - IfcRelConnects (connections)

Global identity and traceability

- Each object has a **GlobalId (persistent across systems)**
- Enables tracking across software platforms and lifecycle phases



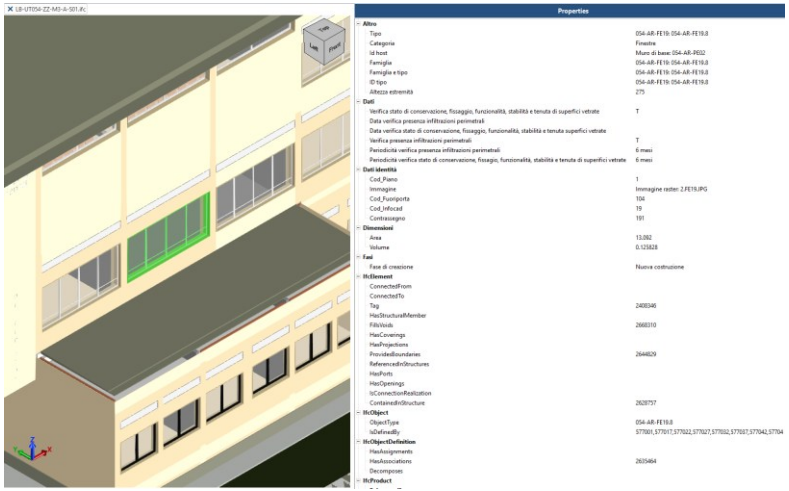
Credit: BIMcert Handbook, Edition 2024

THE IFC DATA MODEL

IFC is more than geometry: it is a **semantic graph** of the building

Each object contains:

- type classification
- material properties
- functional role
- property sets (Psets)

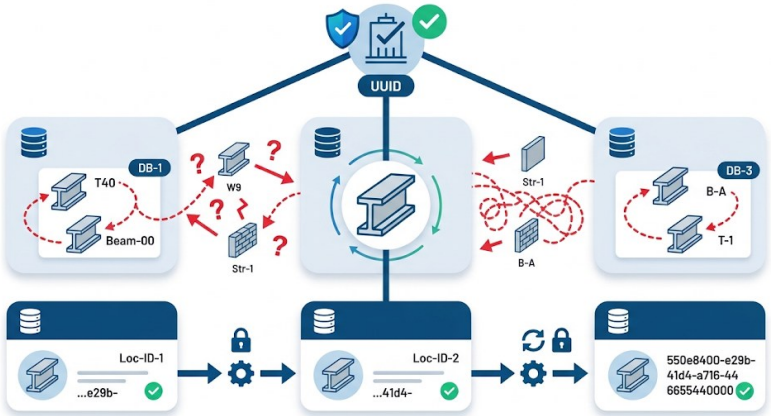


ID UNIQUENESS

IFC (and NGSI-LD) requires that entities have unique ID

How can different systems avoid using the same IDs?

- **NGSI-LD** does not offer a global catalogue; however, ID must be URN/URI, therefore ID can be created based on an owned domain (e.g., `disit.org`)
- **IFC** requires the usage of the `GlobalId`, that is a GUID derived from an UUID identifier at 128 bit coded in 22 characters:
  - probability of collisions is extremely low

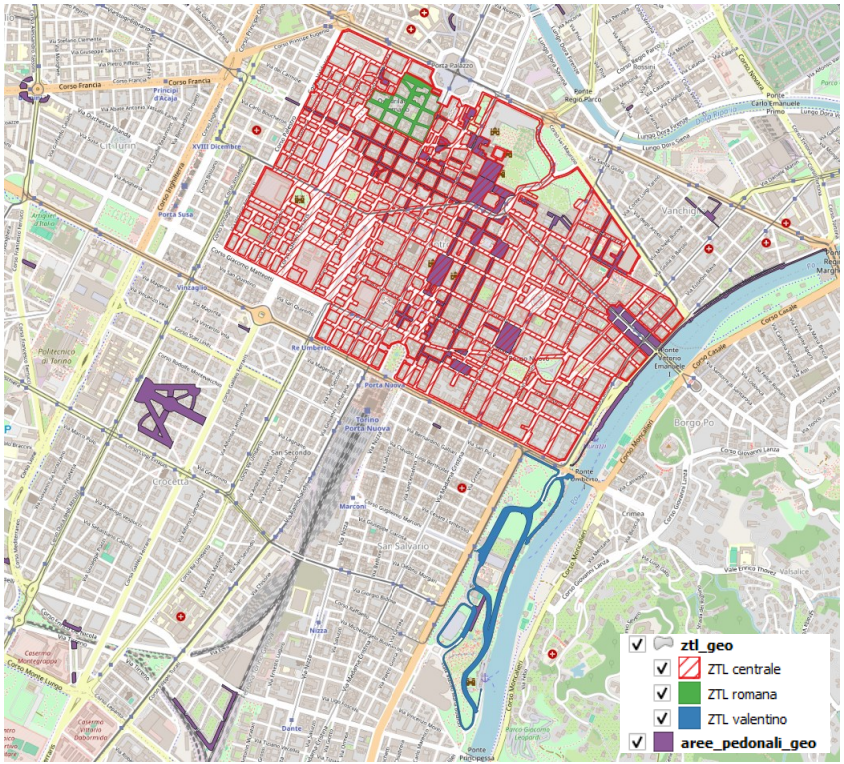


WHAT IS GIS

GIS is a system for **capturing, storing, managing, and analysing geospatial data.** It links **location (where)** with **attributes (what)**

Core idea

- Everything is referenced in a **spatial coordinate system**
- Enables integration of heterogeneous spatial datasets



WHAT IS GIS

Key capabilities

- Spatial visualization (maps, layers, 3D scenes)
- Spatial analysis (distance, overlay, proximity, networks)
- Data integration from multiple sources (sensors, surveys, satellite data)

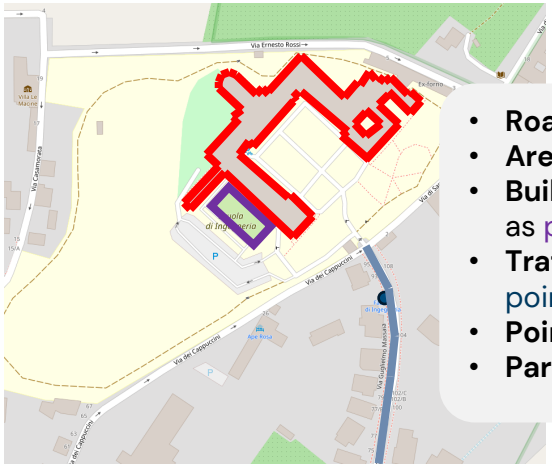
Role in Digital Twins

- Provides the **geospatial reference framework for urban-scale systems**
- Bridges physical space and digital representation

GIS CONCEPTS AND DATA MODELS

Spatial data types

- **Vector data**: points, lines, polygons (buildings, roads, parcels)
- **Raster data**: grids (satellite imagery, elevation models)



- **Roads** represented as line segments
- **Areas** represented as polygons
- **Building ground shapes** represented as polygons or multi-polygons
- **Traffic signs** and **traffic lights** as points
- **Point of Interest** (POI) as point
- **Parking** as points or polygons



GIS CONCEPTS AND DATA MODELS

Attribute model

- Each spatial feature has associated **non-spatial attributes**, e.g., building → height, usage, material, population

Coordinate reference systems (CRS)

- Define how spatial data is positioned on Earth (WGS84, UTM zones, etc.)
- Critical for multi-source data integration

Data layers concept

- GIS organizes information into **thematic layers**
- Layers can be combined for analysis and decision-making



Feature	Value
aree_pedonali_geo	
DENOMINAZI	crocetta
(Derived)	
(clicked coordinate X)	853373
(clicked coordinate Y)	5631151
Area (Cartesian)	49605 m²
Area (Ellipsoidal — EPSG:7030)	49627 m²
Closest X	1394987
Closest Y	4990609
Closest vertex X	853370
Closest vertex Y	5631121
Closest vertex number	114
Feature ID	98
Part number	1
Parts	1
Perimeter (Cartesian)	3069 m
Perimeter (Ellipsoidal — EPSG:7030)	3070 m
Vertices	214
(Actions)	
	View feature form
DENOMINAZI	crocetta
CIRCOSCRIZ	1
SUPERFICIE	49605
ORD_IST	NULL
ID_A_PEDO	99

GIS Data

Geographic Information System (GIS) data:

- Data to represent geographic information
  - Point
  - Line
  - Polygon
  - Multi-polygon

GIS Data

Geographic Information System (GIS) data:

- Data to represent geographic information
- Vector data: each described entity is geo-localized with different shapes
  - Point



GIS Data

Geographic Information System (GIS) data:

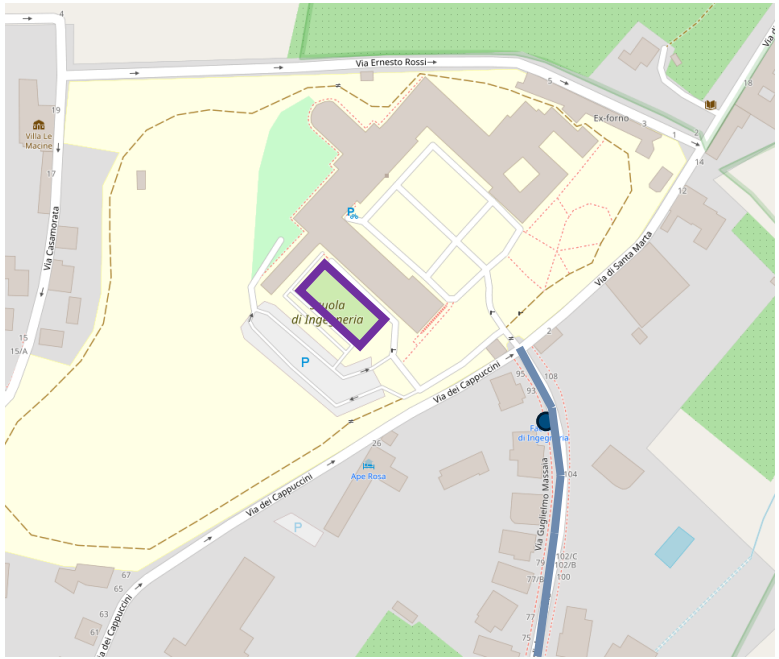
- Data to represent geographic information
- Vector data: each described entity is geo-localized with different shapes
  - Point
  - Line



GIS Data

Geographic Information System (GIS) data:

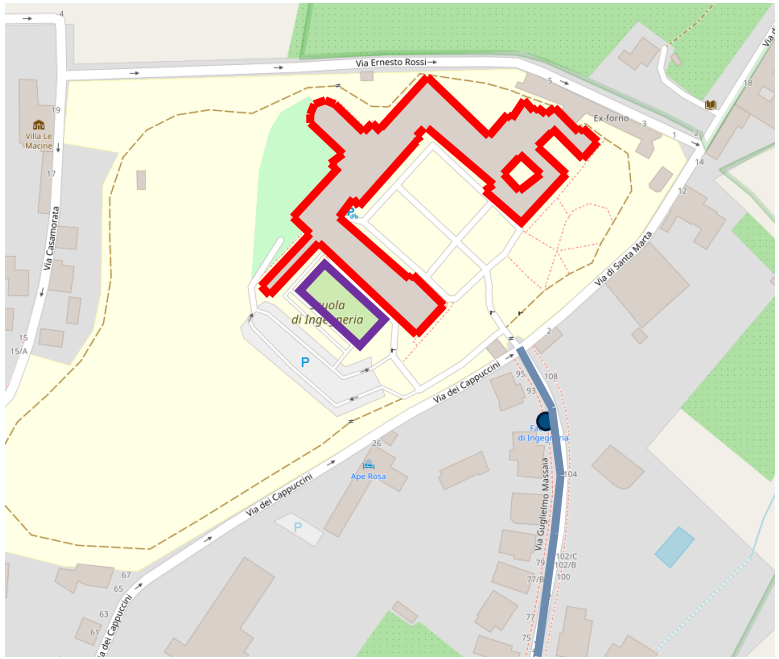
- Data to represent geographic information
- Vector data: each described entity is geo-localized with different shapes
  - Point
  - Line
  - Polygon



GIS Data

Geographic Information System (GIS) data:

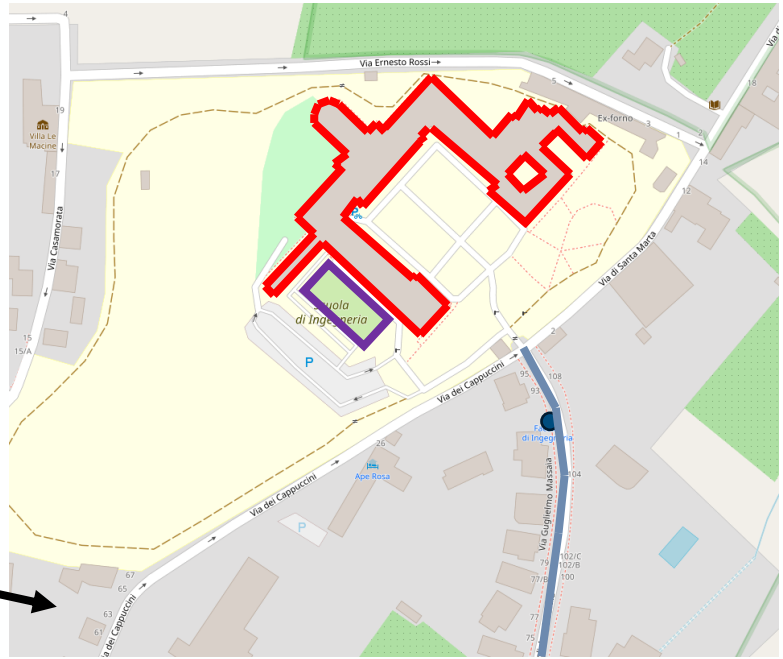
- Data to represent geographic information
- Vector data: each described entity is geo-localized with different shapes
  - Point
  - Line
  - Polygon
  - Multi-polygon



GIS Data

Geographic Information System (GIS) data:

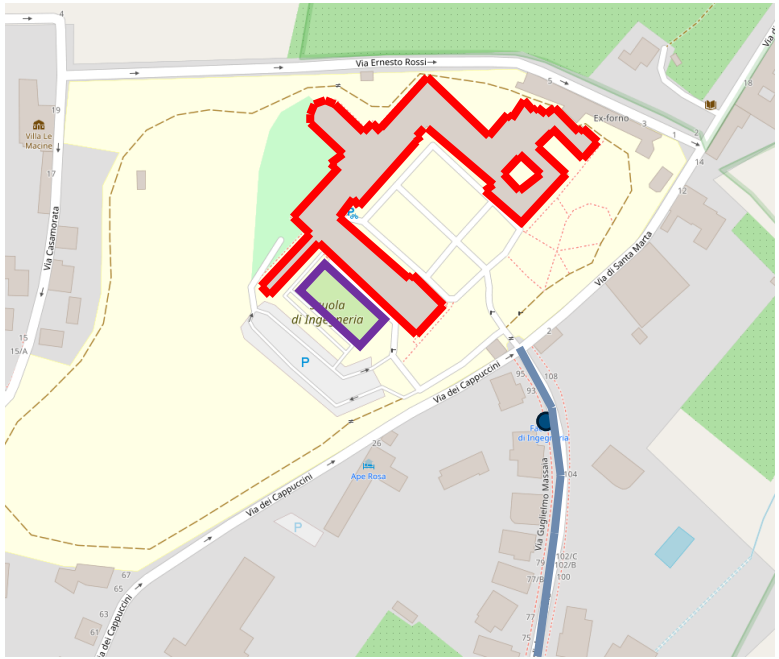
- Data to represent geographic information
- Vector data: each described entity is geo-localized with different shapes
  - Point
  - Line
  - Polygon
  - Multi-polygon
- Raster data: images with localization information



GIS Data – Vectors

GIS data are used to represent and describe the infrastructures

- Roads represented as line segments
- Areas represented as polygons
- Building ground shapes represented as polygons or multi-polygons
- Traffic signs and traffic lights as points
- Point of Interest (POI) as point
- Parking as points or polygons
- …



GIS Data – Raster

GIS data are used to represent and describe the infrastructures

- Ground maps



GIS Data – Raster

GIS data are used to represent and describe the infrastructures

- Ground maps, of different kinds



GIS Data – Raster

GIS data are used to represent and describe the infrastructures

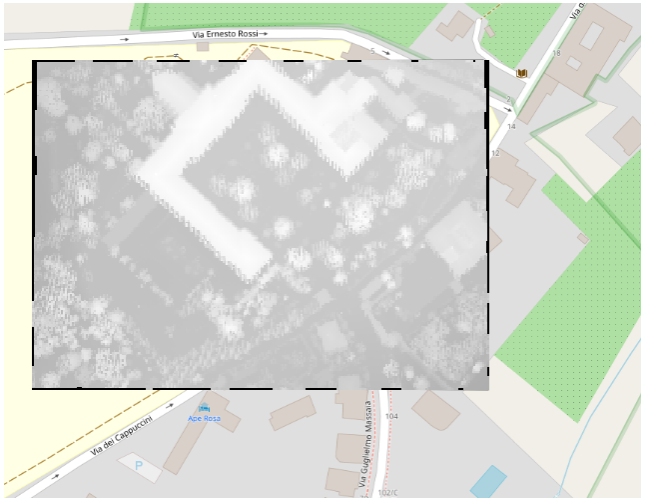
- Ground maps, of different kinds
- Aerial/satellite photos (orthomaps)



GIS Data – Raster

GIS data are used to represent and describe the infrastructures

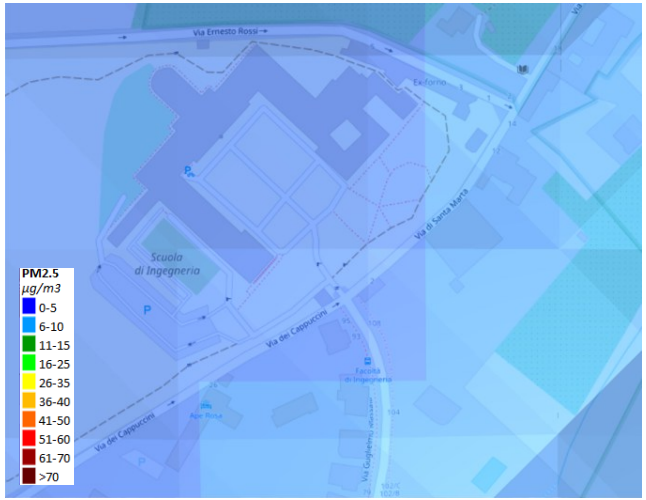
- Ground maps, of different kinds
- Aerial/satellite photos (orthomaps)
- Elevation maps



GIS Data – Raster

GIS data are used to represent and describe the infrastructures

- Ground maps, of different kinds
- Aerial/satellite photos (orthomaps)
- Elevation maps
- Heatmaps

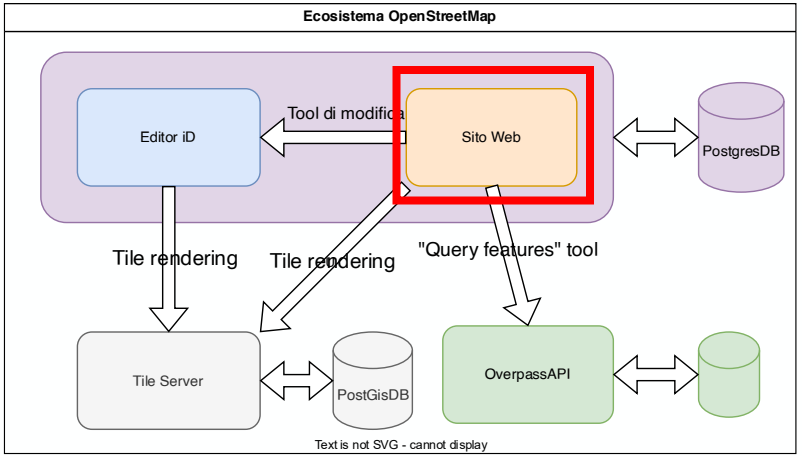


GIS Data – OSM

GIS data can be retrieved from different services

OpenStreetMap (https://www.openstreetmap.org/) is one of the most complete and updated source

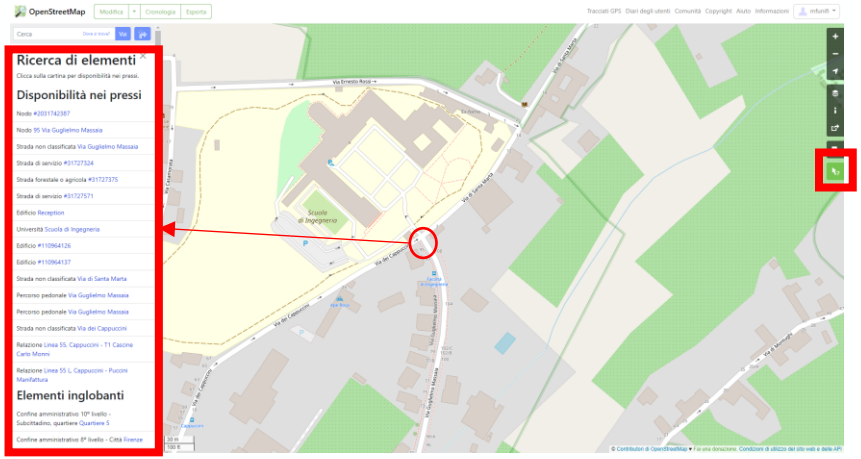


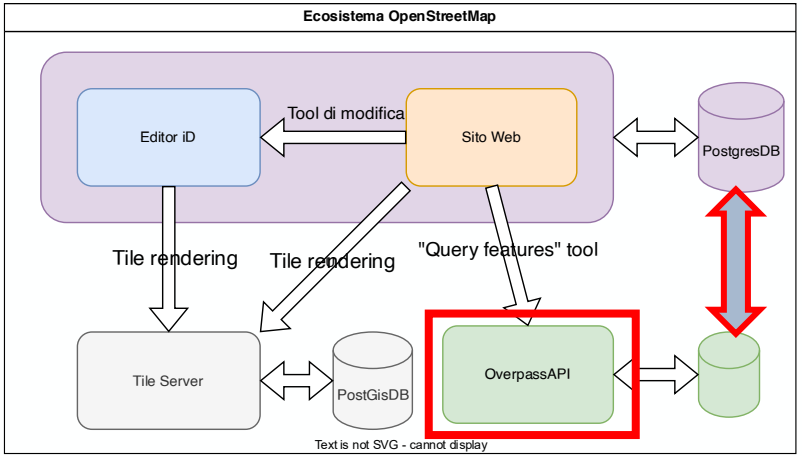


GIS Data – OSM

GIS data can be retrieved from different services

OpenStreetMap (https://www.openstreetmap.org/) is one of the most complete and updated source

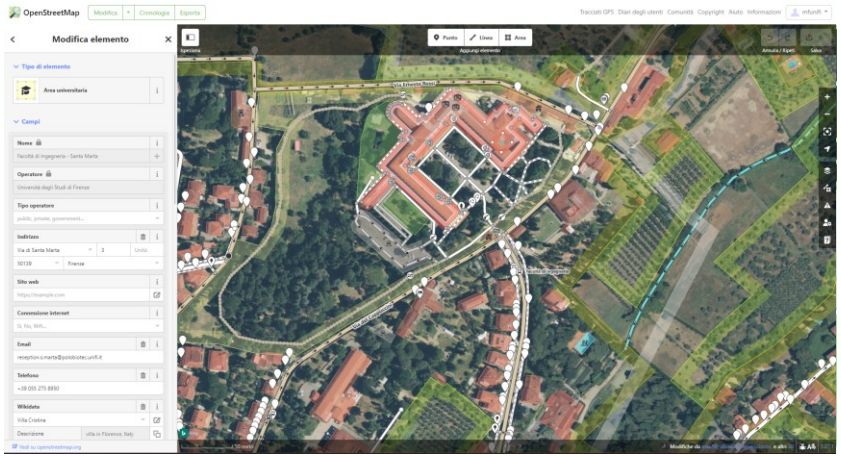


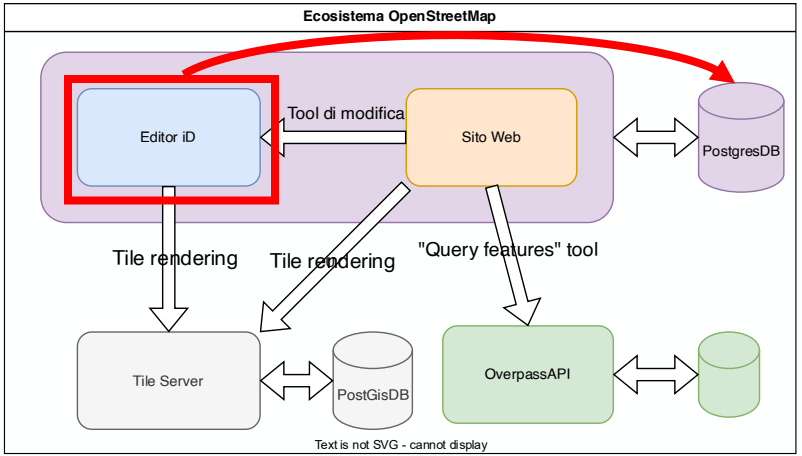


GIS Data – OSM

GIS data can be retrieved from different services

OpenStreetMap (https://www.openstreetmap.org/) is one of the most complete and updated source

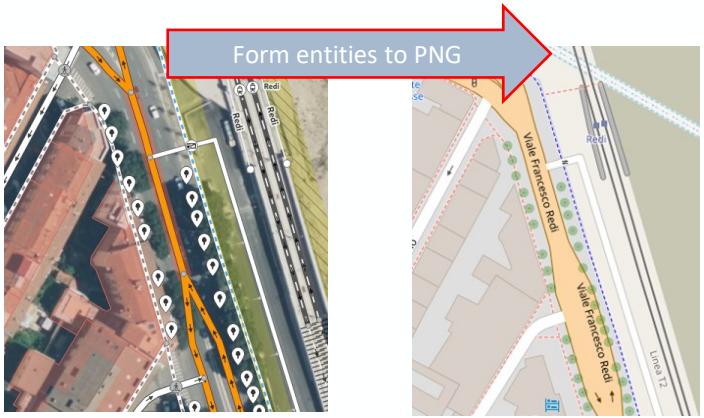


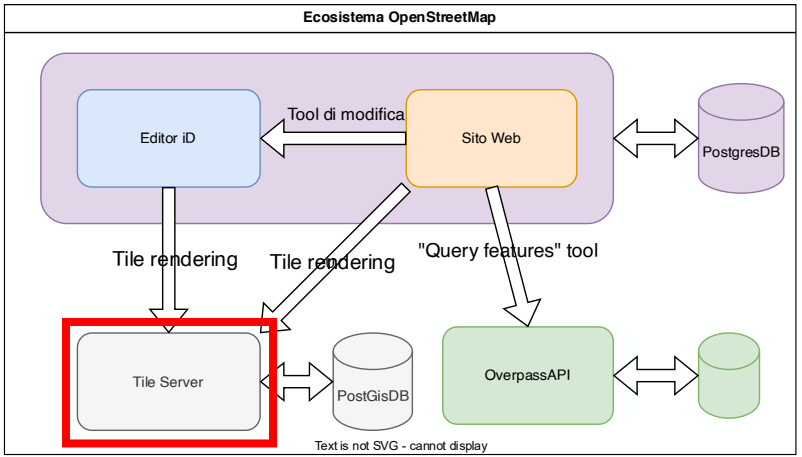


GIS Data – OSM

GIS data can be retrieved from different services

OpenStreetMap (https://www.openstreetmap.org/) is one of the most complete and updated source

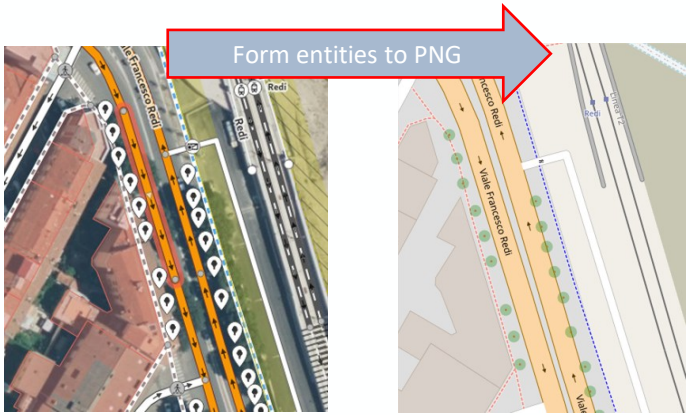


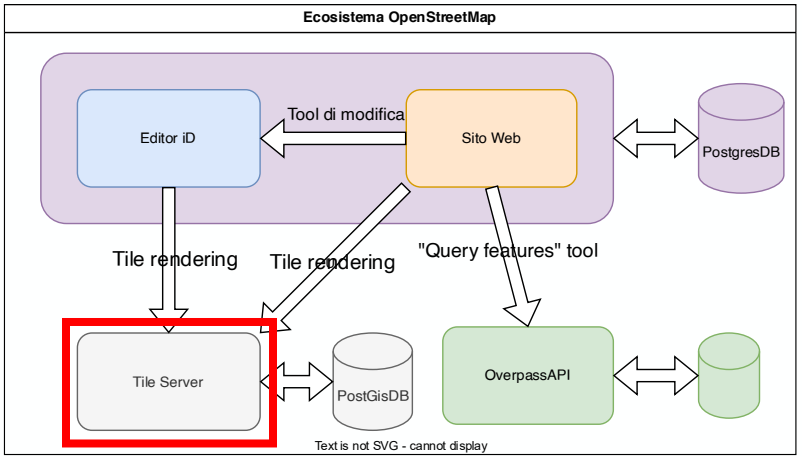


GIS Data – OSM

GIS data can be retrieved from different services

OpenStreetMap (https://www.openstreetmap.org/) is one of the most complete and updated source





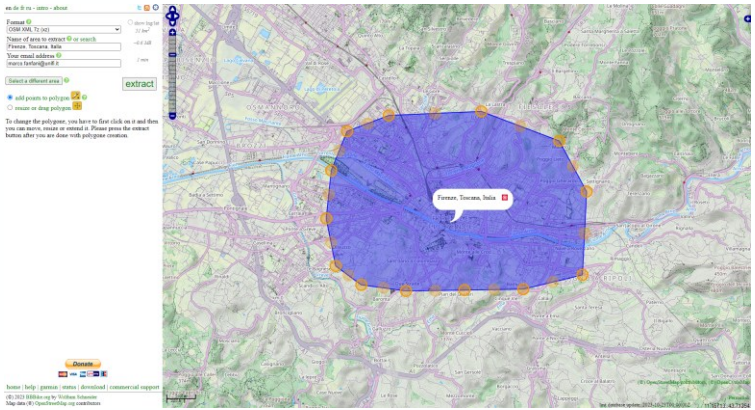
GIS Data – OSM

GIS data can be retrieved from different services

OpenStreetMap (https://www.openstreetmap.org/) is one of the most complete and updated source

Data from OSM DB can be exported using web services:

- Geofabrik (https://download.geofabrik.de/)
- HOT Export Tool (https://export.hotosm.org/)
- BBBike (https://download.bbbike.org/osm/)



GIS Data – OSM

OSM data can be retrieved directly from the PostgreSQL DB

This relational DB is composed by tables representing

- Nodes (i.e., points)
- Ways (i.e., lines and polygons)
- Relations (i.e., multilines and multipolygons)

OSM can be deployed on-premise, see https://github.com/openstreetmap/openstreetmap-website

GIS Data – Vector formats

OSM data can be obtained in different formats

The preferred format is the XML OSM standard. It includes descriptions of

- Nodes
- Ways
- Relations

```xml
<node id="1533919" lat="43.7958032" lon="11.2447127" version="1">
    <tag k="crossing" v="zebra"/>
    <tag k="crossing:island" v="no"/>
    <tag k="crossing:markings" v="yes"/>
    <tag k="highway" v="crossing"/>
    <tag k="tactile_paving" v="no"/>
</node>
```

```xml
<way id="111566019" version="1">
    <nd ref="266934805"/>
    <nd ref="8256676004"/>
    <nd ref="2473224"/>
    <tag k="cycleway" v="track"/>
    <tag k="highway" v="primary"/>
    <tag k="lanes" v="3"/>
    <tag k="name" v="Viale Giacomo Matteotti"/>
    <tag k="oneway" v="yes"/>
    <tag k="surface" v="asphalt"/>
</way>
```

```xml
<relation id="9128948" version="1">
    <member type="way" ref="28618378" role="outer"/>
    <member type="way" ref="656322769" role="inner"/>
    <tag k="addr:city" v="Firenze"/>
    <tag k="addr:housename" v="Santa Marta"/>
    <tag k="addr:housenumber" v="3"/>
    <tag k="addr:postcode" v="50139"/>
    <tag k="addr:street" v="Via di Santa Marta"/>
    <tag k="amenity" v="university"/>
    <tag k="building" v="university"/>
    <tag k="building:levels" v="3"/>
    <tag k="building:levels:underground" v="1"/>
```

GIS Data – Vector formats

Alternatively, OSM data can be downloaded as **GeoJSON** files

GeoJSON augment the standard JSON format with **geographical features**


```
{
  "type": "Feature",
  "properties": {
    "osm_id": "9128948",
    "osm_type": "relation",
    "building": "university",
    "name": "Facoltà di Ingegneria - Santa Marta",
    "type": "multipolygon"
  },
  "geometry": {
    "type": "MultiPolygon",
    "coordinates": [
      [
        [
          [
            11.252543706842182,
            43.79816670104470 7
          ],
          [
            11.252441106842232,
            43.798236701044694
          ],
          [
            11.252745706842136,
            43.798472201044646
          ],
          [
            11.252709606842153,
            43.79849500104465
          ],
          [
            11.252774806842131,
            43.798546501044655
          ],
          [
            11.25257860684223,
            43.798678101044651
          ],
          [
            11.252561106842244,
            43.79869930104465
          ],
```

GIS Data – Vector formats

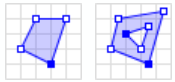
GeoJSON can define several geometric structure


```
"geometry": {
  "type": "Point",
  "coordinates": [102.0, 0.5]
}
```



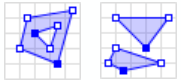
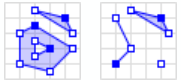
```
"geometry": {
  "type": "LineString",
  "coordinates": [
    [102.0, 0.0],
    [103.0, 1.0],
    [104.0, 0.0],
    [105.0, 1.0]
  ]
}
```

```
"geometry": {
  "type": "Polygon",
  "coordinates": [
    [
      [100.0, 0.0],
      [101.0, 0.0],
      [101.0, 1.0],
      [100.0, 1.0],
      [100.0, 0.0]
    ]
  ]
}
```



GIS Data – Vector formats

GeoJSON can define several geometric structure

- Point
- Linestring 
- Polygon 
- Multipoint 
- MultiLinestring 
- Multipolygon 
- GeometryCollection 

GIS Data – Vector formats

GeoJSON use the EPGS:4326 – WGS84 coordinate reference system (CRS), defining the coordinate of point in latitude and longitude

When using different CRS, this information must be specified

```
"crs": {
  "type": "name",
  "properties": {
    "name": "urn:ogc:def:crs:EPSG::3003"
  }
},
```

GIS Data – Vector formats

OSM data can be retrieved also as **Esri Shapefile** format

- Developed and regulated by Esri as a **mostly open** specification for data interoperability
- It can represent the entities already seen (points, lines, polygons, …)
- It is composed by several mandatory and optional files
  - `.shp` — shape format;
  - `.shx` — shape index format;
  - `.dbf` — attribute format, columnar attributes for each shape;
  - `.prj` — projection description

GIS Data – Vector formats

Another format to represent geometric spatial objects is the **Well-Known Text (WKT)** format.

Similarly to GeoJSON, WKT can encode spatial entities like

Entity	Syntax
Point	`POINT(1 2)`
LineString	`LINESTRING(1.5 2.45, 3.21 4)`
Polygon	`POLYGON((1 2,1 4,3 4,3 2,1 2))` `POLYGON((0.5 0.5,5 0,5 5,0 5,0.5 0.5), (1.5 1,4 3,4 1,1.5 1))`
MultiPoint	`MULTIPOINT(0 0,1 1)`
MultilineString	`MULTILINESTRING((0 0,-1 -2,-3 -4),(2 3,3 4,6 7))`
MultiPolygon	`MULTIPOLYGON(((0 1,3 0,4 3,0 4,0 1)), ((3 4,6 3,5 5,3 4)), ((0 0,-1 -2,-3 -2,-2 -1,0 0)))`
GeometryCollection	`GEOMETRYCOLLECTION(POINT(5 8), LINESTRING(-1 3,1 4))`
Coordinate Ref. System	`SRID=<ID>;<ENTITY>` (e.g., `SRID=4326;POINT(11.255 43.77)`)

GIS Data – Raster formats

Raster data, i.e., images, requires different formats

Most common formats include

- PNG
- GeoTIFF
- Esri grid ASCII (ASC)

GIS Data – PNG

Portable Network Graphics (PNG) is a raster-graphics file format that supports lossless data compression

- PNG is often the best raster format for maps, since the lossless compression keeps text and line work readable by preventing the compression artifacts (unlike JPEG)
- PNG can handle transparency (useful to indicate no-data values)
- PNG is widely used in many applications, in particular for web applications
- However, **PNG does not embed geographic information!**

GIS Data – GeoServer

Maps represented as PNG are usually served by GIS servers such as **GeoServer.** GeoServer supports the Open Geospatial Consortium (OGC) **Web Map Service (WMS) standard**, which allows clients to request map images over HTTP/HTTPS.

- The client sends a request specifying the
  - geographic bounding box (BBOX),
  - coordinate reference system (CRS),
  - image dimensions (width and height),
  - requested map layers,
  - desired image format (e.g., PNG).
- The server renders the requested map and returns the resulting image.
- Its geographic extent and spatial reference are defined by the request parameters and **must be retained by the client to correctly position the image on a map**.

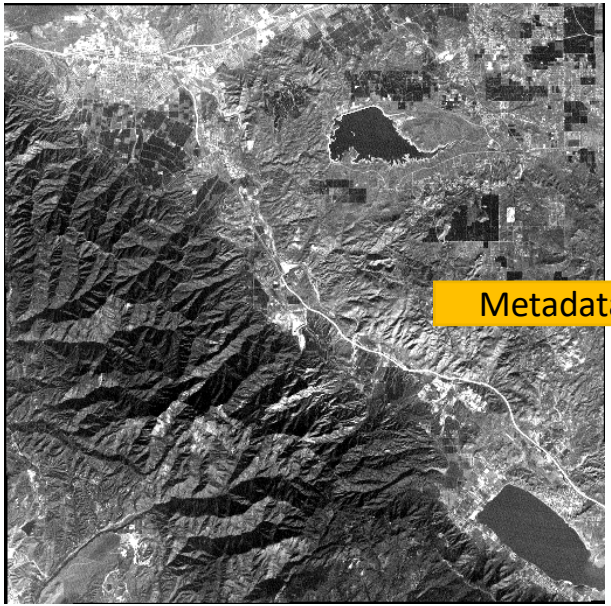


Example REST call:

```
https://wmsserver.snap4city.org/geoserver/Snap4City/wms?service=WMS&request=GetMap&layers=dtm_orografico_SuperRes_epgs4326_RGB&format=image%2Fpng&transparent=true&width=256&height=256&bbox=11.24725341796875,43.76911045236617,11.25,43.7710938177564 9
```

GIS Data – GeoTIFF

GeoTIFF is a public domain metadata standard which allows georeferencing information to be embedded into a TIFF file.




```
Driver: GTiff/GeoTIFF
Files: cea.tif
Size is 514, 515
Coordinate System is:
PROJCRS[
  …
]
Data axis to CRS axis mapping: 1,2
Origin = (-28493.166784412522247,4255884.543802191503346)
Pixel Size = (60.022136983193739,-60.022136983193739)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  INTERLEAVE=BAND
Corner Coordinates:
Upper Left  (  -28493.167, 4255884.544) (117d38'27.05"W, 33d56'37.74"N)
Lower Left  (  -28493.167, 4224973.143) (117d38'27.05"W, 33d39'53.81"N)
Upper Right (    2358.212, 4255884.544) (117d18'28.38"W, 33d56'37.74"N)
Lower Right (    2358.212, 4224973.143) (117d18'28.38"W, 33d39'53.81"N)
Center      (  -13067.478, 4240428.844) (117d28'27.71"W, 33d48'15.38"N)
Band 1 Block=514x15 Type=Byte, ColorInterp=Gray
```

GIS Data – ASC

Esri grid ASCII (ASC) is a text format to represent matrix data with geographic information

ASC is composed by two main part

- Header with metadata
- Cell values

Note that, there is not CRS information!

```
NCOLS 1640
NROWS 1240
XLLCENTER 1675980.500
YLLCENTER 4846780.500
CELLSIZE 1
NODATA_VALUE -9999.000
 41.236 41.226 41.226 41.226 41.256 41.216 41.206 41.156 …
 41.226 41.176 41.176 41.176 41.196 41.206 41.226 41.227 …
 41.287 41.307 41.287 41.257 41.287 41.307 41.317 41.317 …
 41.307 41.397 41.367 41.397 41.407 41.427 41.487 41.467 …
 41.447 48.407 51.757 52.207 47.947 49.867 46.577 46.197 …
 …
```

THE EVOLUTION FROM 2D GIS TO 3D GIS

Urban Digital Twins require moving beyond flat maps.

Traditional GIS
- Primarily 2D representations
- Buildings represented as footprints

Need for 3D Models
- Urban planning
- Shadow simulations
- Wind/flooding analysis
- TLC planning

Emergence of 3D GIS Standards
- CityGML
- IndoorGML
- 3D Tiles

CITYGML: SEMANTIC 3D CITY MODEL

CityGML is an **open standard** for 3D city modeling developed by OGC (Open Geospatial Consortium). It extends GIS from 2D maps to **semantically rich 3D urban objects**

Core concept: Represents cities as **objects with geometry + semantics + topology** with focus on

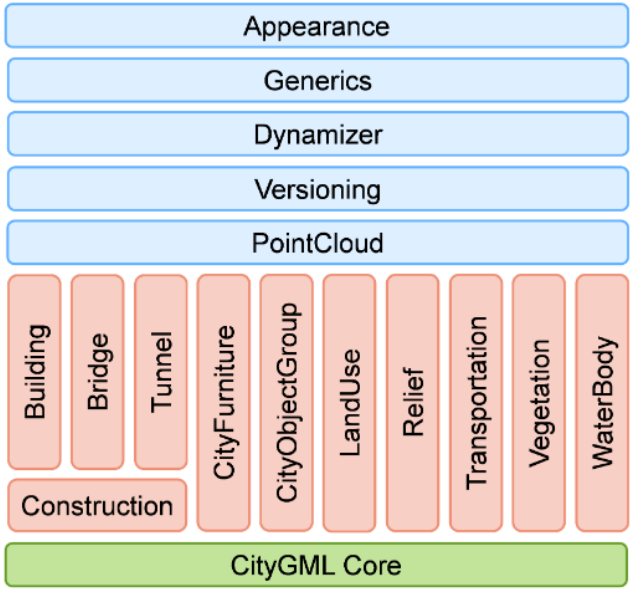
- buildings,
- terrain,
- transportation,
- vegetation,
- infrastructure



CITYGML THEMATIC MODULES

A city is represented as a collection of semantically classified objects. CityGML adopts a **modular architecture**. A common **Core Module** provides fundamental concepts, while thematic modules describe specific urban entities.

- **Building Module:** Buildings and building parts
- **Transportation Module:** Roads, Railways, Squares
- **Vegetation Module:** Trees, Green areas
- **Water Module:** Rivers, Lakes, Coastal areas
- **Relief Module:** Terrain and elevation



Credit: https://docs.ogc.org/guides/20-066.html

UNDERSTANDING LEVELS OF DETAIL (LOD)

LOD0: Terrain and regional models

LOD1: Simple building blocks

LOD2: Roof structures and basic architectural features

LOD3: Detailed external architecture

LOD4: Interior spaces



Credit: Biljecki et al. An improved LOD specification for 3D building models. 2016

CITYGML: DATA FORMAT

CityGML is implemented as an application schema of GML (an **XML grammar** developed by the OGC for geographic information). Every city object is represented as an XML element with **geometry** and **attributes**.



- Human-readable structure
- Platform-independent representation
- Self-describing data through tags
- Strong interoperability between GIS systems



- Large file sizes
- Verbose syntax
- Computationally expensive parsing for large city models

CITYGML: DATA FORMAT

- XML tags carry semantic meaning
- Objects are uniquely identified through gml:id
- Geometry and semantics coexist in the same structure
- Hierarchical organization reflects real-world relationships

```
<core:cityObjectMember>
  <bldg:Building gml:id="B001">

    <gml:name>City Hall</gml:name>

    <bldg:function>administration</bldg:function>

    <bldg:boundedBy>
      <bldg:WallSurface>
        ...
      </bldg:WallSurface>
    </bldg:boundedBy>

  </bldg:Building>
</core:cityObjectMember>
```

CITYGML: DATA FORMAT

Geometries are described explicitly through **coordinates that are stored directly in XML**

Supports **2D and 3D geometries**

Can represent:

- Points
- Curves
- Surfaces
- Solids

Limitation: Large buildings may require thousands of coordinates, making **XML files extremely large**

```
<gml:Polygon>
   <gml:exterior>
      <gml:LinearRing>
         <gml:posList>
            0 0 0
            10 0 0
            10 10 0
            0 10 0
            0 0 0
         </gml:posList>
      </gml:LinearRing>
   </gml:exterior>
</gml:Polygon>
```

LIMITATIONS OF CITYGML FOR DIGITAL TWINS

Static by Design:

- Intended for city representation
- Not for real-time updates

Large XML Files:

- High storage requirements
- Slow parsing and processing

Limited Temporal Dimension:

- Difficult management of continuously changing data

Limited Sensor Integration:

- IoT data requires external platforms

BIM Integration Challenges:

- IFC contains richer engineering information
- Semantic mapping remains complex

WHY XML BECAME A BOTTLENECK

Strengths

- Standardized
- Interoperable
- Extensible
- Human-readable

Weaknesses

- Very verbose
- Large storage requirements
- Slow web transmission
- Inefficient visualization pipelines

Consequences

The emergence of alternative representations:

- 3D Tiles
- glTF-based visualization pipelines

WHAT IS 3D TILES

3D Tiles is an open specification developed by the Cesium ecosystem and now maintained by the Open Geospatial Consortium.

It is designed for **efficient streaming** and **visualization of massive heterogeneous 3D geospatial datasets** and optimized for web-based rendering and real-time exploration

Supported content

- 3D city models
- BIM models
- Point clouds
- Photogrammetry meshes
- Terrain datasets

Main objective

- Deliver only the data needed for the current viewpoint
- Minimize memory usage and network traffic
- Enable visualization of city- and country-scale models

HOW 3D TILES WORK

Tile-based organization

- The dataset is divided into hierarchical spatial tiles
- Each tile contains a subset of the scene
- Tiles are loaded dynamically according to camera position

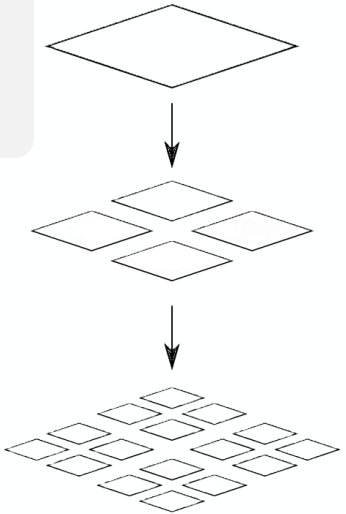
Level of Detail (LOD)

- Distant objects use simplified representations
- Nearby objects use more detailed geometry
- Continuous adaptation during navigation

Hierarchical structure

Hierarchical structure

```
Root Tile
 ├─ Tile A
 │    ├─ Tile A1
 │    └─ Tile A2
 └─ Tile B
      ├─ Tile B1
      └─ Tile B2
```



OBJECT IDENTIFICATION IN 3D TILES

The Challenge

- A 3D Tiles dataset may contain thousands of objects within the same tile
- Rendering systems need a mechanism to identify individual features
- Geometry alone does not provide object identity

Batch IDs / Feature IDs

- A unique identifier assigned to each feature within a tile
- Associates geometric primitives with a logical object
- Enables:
  - feature picking
  - object highlighting
  - metadata retrieval
  - filtering and styling

Feature ID is stored as:
- **per-vertex attribute**
- or **per-primitive attribute**
- or **per-instance attribute**

Each vertex (or triangle) carries:

```
Triangle 1 → featureId = 12
Triangle 2 → featureId = 12
Triangle 3 → featureId = 13
```

FEATURE IDS AND METADATA ASSOCIATION

`click → raycast → triangle index → featureId → metadata lookup`

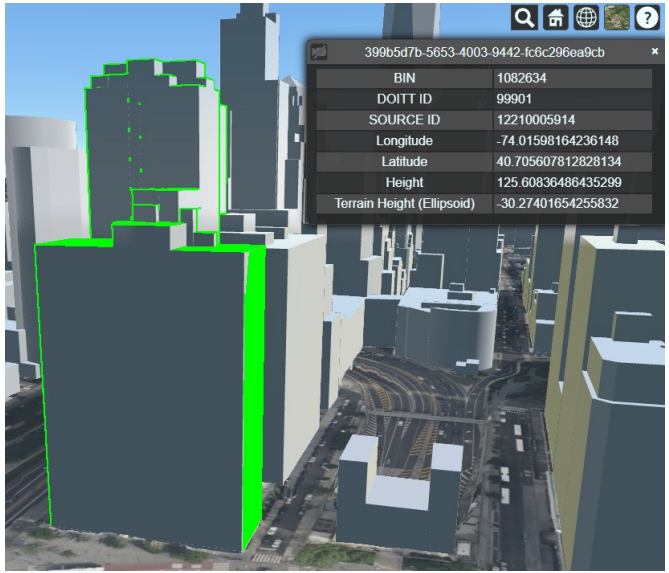
Linking Geometry and Information

Each rendered object can be associated with metadata through its Feature ID.

Benefits

- Attribute querying
- Object selection
- Dynamic styling
- Integration with external databases

Key Message: *The geometry remains optimized for rendering, while metadata are stored separately and linked through Feature IDs.*



Credit: https://sandcastle.cesium.com/index.html?id=3d-tiles-feature-picking

LIMITATIONS OF FEATURE-LEVEL SEMANTICS

Feature IDs Provide

✓ Object identification

✓ Attribute association

✓ Interactive querying

✓ Object styling



Feature IDs Not Provide

✕ Rich semantic hierarchies

✕ Object relationships

✕ Topological structures

✕ Urban ontologies

✕ Building decomposition



glTF/GLB

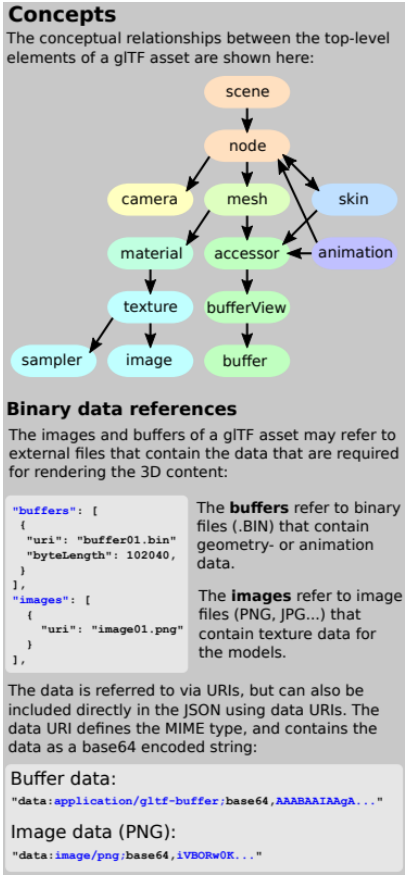
glTF (GL Transmission Format) is an **open standard for efficient transmission of 3D models,** developed by the Khronos Group, often called the **"JPEG of 3D"** for real-time graphics

Main purpose

- Designed for **fast loading and rendering in web and mobile environments**
- Optimized for GPU consumption (not for editing or authoring)

Two formats

- **.gltf** → JSON + external resources (textures, buffers)
- **.glb** → binary packed version (single file)



Credit: https://www.khronos.org/files/gltf20-reference-guide.pdf

glTF/GLB

glTF/GLB is not a modelling standard, but a highly efficient delivery format for real-time 3D visualization across heterogeneous platforms.

1. Real-time optimized format

- Designed for **GPU consumption**
- Minimal CPU parsing overhead
- Direct mapping to graphics pipelines (WebGL / Vulkan / Metal)

2. Fast loading and transmission

- Compact binary encoding (.glb)
- Efficient use of buffers and indexing
- Optional compression (Draco, texture compression)

3. Standardized scene representation

- Open standard maintained by Khronos Group
- Widely supported across engines

4. Portable across platforms

- Web, mobile, desktop, XR
- No vendor lock-in
- Same asset reusable in multiple environments

5. Physically Based Rendering (PBR) support

- Realistic materials out of the box
- Metallic-roughness workflow
- Consistent appearance across renderers

6. Suitable for large-scale visualization

- City-scale 3D visualization via instancing
- Works well as output format for BIM, GIS-derived, Digital twin front-ends

glTF/GLB SEMANTIC LIMIT

But glTF does NOT provide natively:

- ✕ No urban semantics
- ✕ No object ontology
- ✕ No spatial relationships
- ✕ No domain concepts (building, road, sensor)

glTF describes how something looks, not what something is!

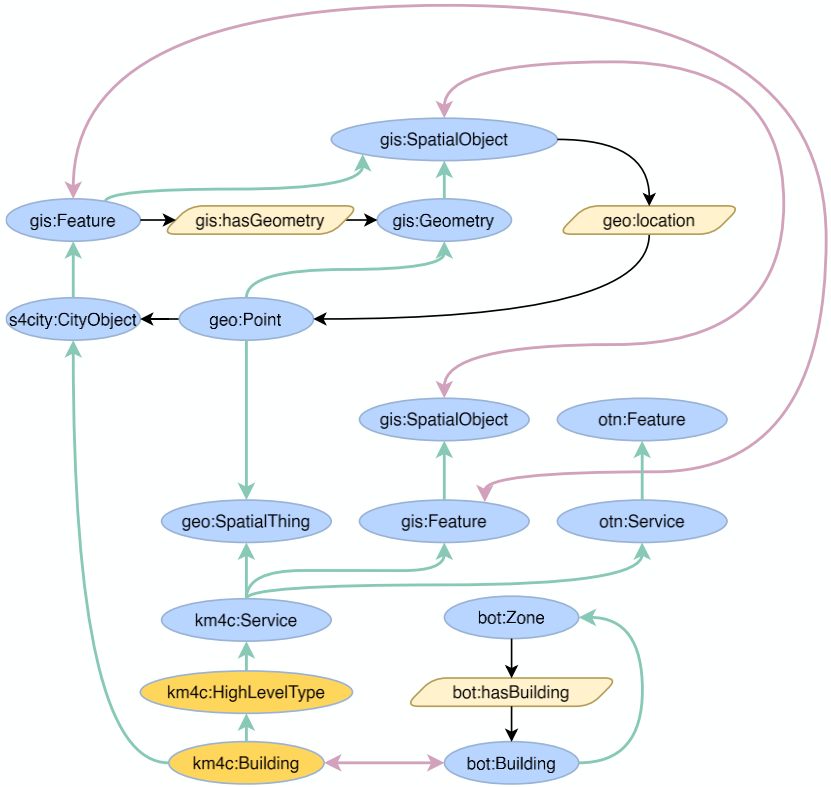
COMPARISON OF 3D FORMATS FOR DIGITAL TWINS

Aspect	IFC	CityGML	3D Tiles	glTF / GLB
Primary goal	BIM / engineering model	Urban semantic model	Streaming & visualization	Real-time rendering format
Scale	Building	City / region	City-scale streaming	Object / scene visualization
Geometry	Very detailed	Detailed + structured	Optimized meshes / tiles	GPU-ready meshes
Semantics	Very rich (engineering)	Rich (urban objects)	Limited (metadata-based)	Minimal (optional metadata)
Relationships	Explicit (building graph)	Explicit (urban ontology)	Very limited	None
Data model	Object-based schema	XML/GML schema	Tile hierarchy	Scene graph
Strength	Engineering accuracy	Urban semantics	Scalability + streaming	Rendering performance
Weakness	Heavy, not scalable	Verbose, heavy XML	Weak semantics	No semantic model

SNAP4CITY SOLUTION

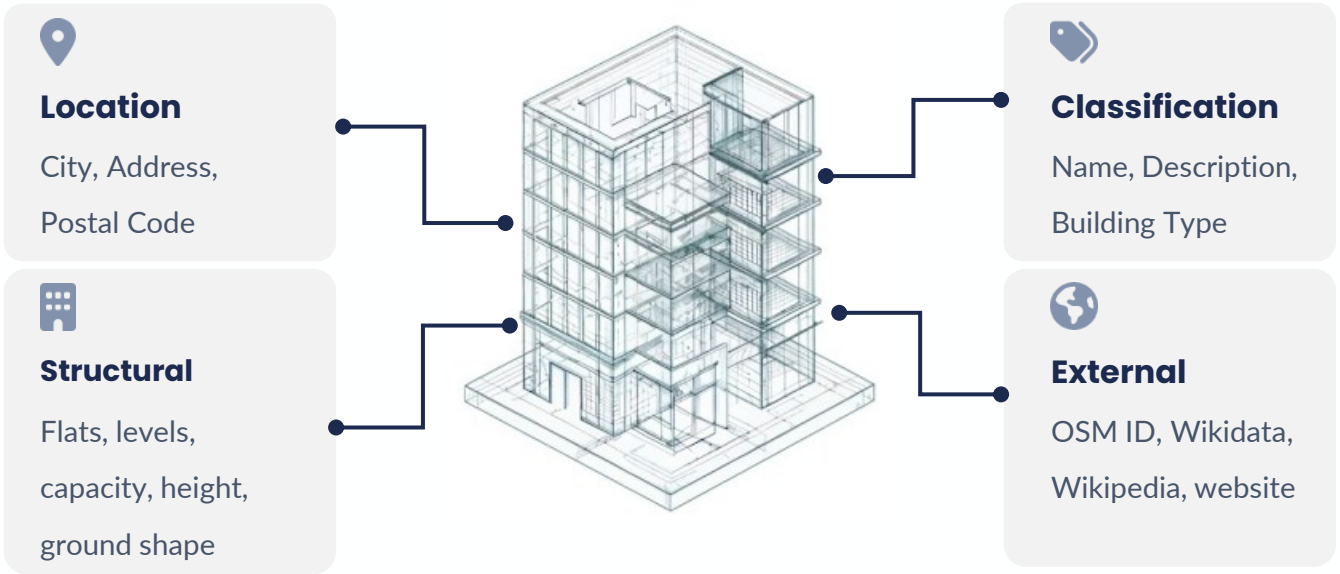
Exploit the advantage of glTF/GLB 3D models and enrich them with semantic by linking to ontologies

- each building is defined as an entity in the KB (based on the KM4City ontology) and described with a detailed data model
- multiple 3D models, including BIM, can be associated with a KB building entity



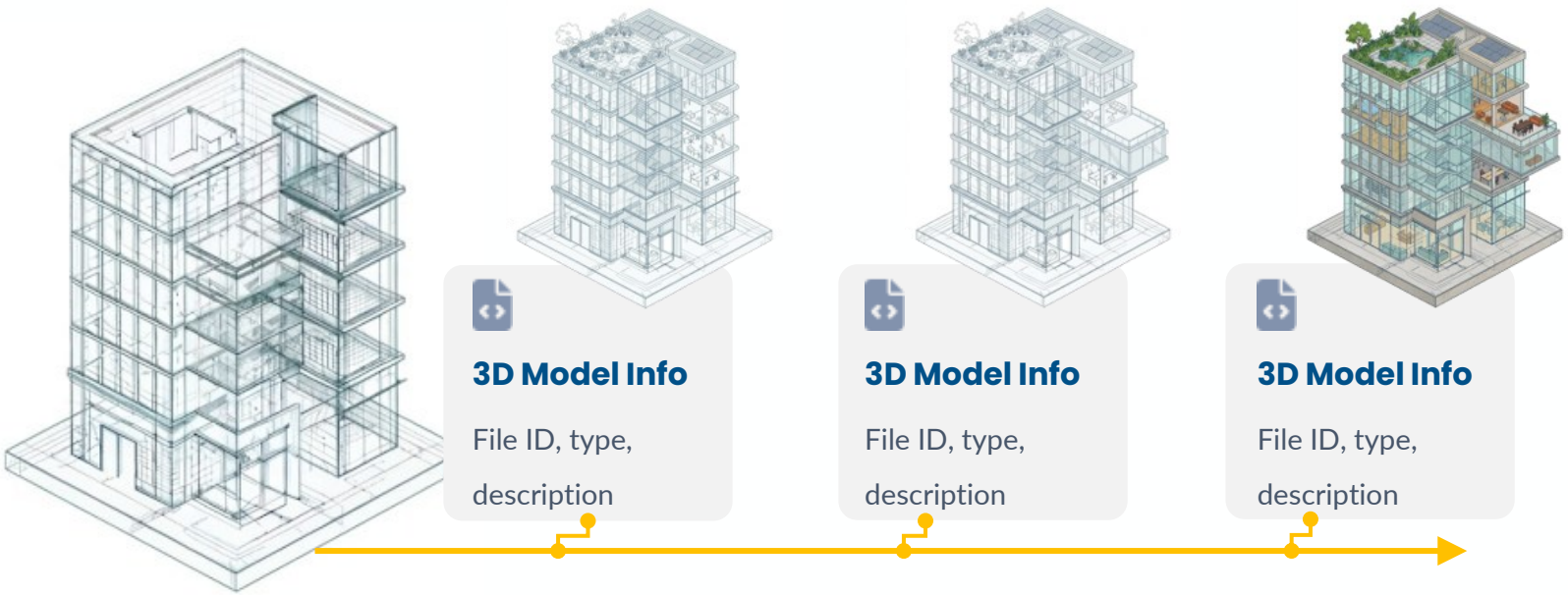
BUILDING DATA MODEL

Each building is enriched with additional data following a specific data model



BUILDING DATA MODEL

Multiple 3D models can be associated to the same building entity



THE BIM-GIS INTEGRATION GAP

Bridging building-level micro-data with city-level macro-context

- micro-scale BIM (building components)
- macro-scale GIS (urban environment)

Scale Disparity: Managing the detail of thousands of BIM models within a single GIS environment

Incomplete Transfers: Many elements of IFC buildings cannot be semantically transferred into GIS models (e.g. CityGML) without loss

BIM–GIS INTEGRATION: WHY IT IS DIFFICULT

Integrating IFC (BIM) and GIS is not a simple conversion problem, because they represent **different conceptualizations of the built environment**.

Different modelling paradigms

BIM
- Object-centric engineering model
- Fine-grained components (walls, beams, slabs, MEP systems)
- Strong internal structure and dependencies

GIS
- Urban object model
- Focus on buildings as holistic entities
- Multi-Level of Detail (LoD0–LoD4)

Semantic mismatch

- Same concept ≠ same meaning, e.g.:
  - "Building" in IFC = engineering system
  - " Building" in CityGML = spatial urban object
- IFC contains concepts with no GIS equivalent
- GIS model introduces abstractions not present in BIM

Geometric and LoD conflicts

- IFC: detailed solid-based geometry
- GIS models: surface-based generalized geometry
- Conversion requires:
  - simplification
  - abstraction
  - loss of detail

BIM–GIS INTEGRATION: MODEL TRANSFORMATION

Complex synchronization: BIM e GIS must remain synchronized

Data Expansion: File sizes often expand significantly during conversion process

Computational Burden: Processing high-fidelity models necessitates high-performance computing and rendering. Light 3D models are preferable, also for web transmission and visualization

Case Example BIM-CityGML:

BIM (IFC): 17,824 KB.

CityGML LOD4: 310,055 KB

CONSEQUENCES OF BIM–GIS FUSION

What is gained

- Urban-scale consistency
- Spatial analytics capability
- Standardized city representation

Gain	Loss
Urban scalability	Engineering detail
GIS compatibility	BIM semantics
Visualization consistency	Structural richness

What is lost in integration

- Engineering-level semantics (materials, systems, functions)
- Internal building structure (MEP, components)
- Precise geometric fidelity

BIM–GIS integration is not neutral: it forces a semantic simplification of the building model.

ONTOLOGY-BASED INTEGRATION

Instead of converting data we can align meanings. IFC becomes a **building ontology** and put in relation with an **urban ontology** (e.g., KM4City). Integration happens at conceptual level:

- classes
- properties
- relationships

Example mapping (conceptual)

- IfcBuilding → km4c:Building
- IfcWall → km4c:Wall
- IfcDoor → km4c:Entry

ONTOLOGY-BASED INTEGRATION

3D data models remain independent but semantically connected through a shared ontology layer.

- BIM preserves full engineering detail, i.e., structural components, materials, systems
- GIS models enable scalable urban representation and analysis

Semantic integration layer

- A **knowledge graph (e.g. KM4City)** acts as semantic backbone
- Each building (and urban object) becomes a **unique semantic entity**
- IFC and GIS models are **linked**, not merged

Reasoning and inference

- **Concept linking** enables:
  - cross-domain queries (BIM ↔ GIS ↔ IoT)
  - semantic enrichment of geometry
  - automatic inference on relationships
- Supports **advanced digital twin functionalities** beyond visualization