

POLITECNICO DI TORINO

I Facoltà di Ingegneria  
Corso di Laurea Magistrale in Ingegneria Edile

Tesi di Laurea Magistrale

**Implementation of BIM methodology for district  
data visualization**



**Relatore:**

prof.ssa Anna Osello

**Candidato:**

Fabrizio Provera

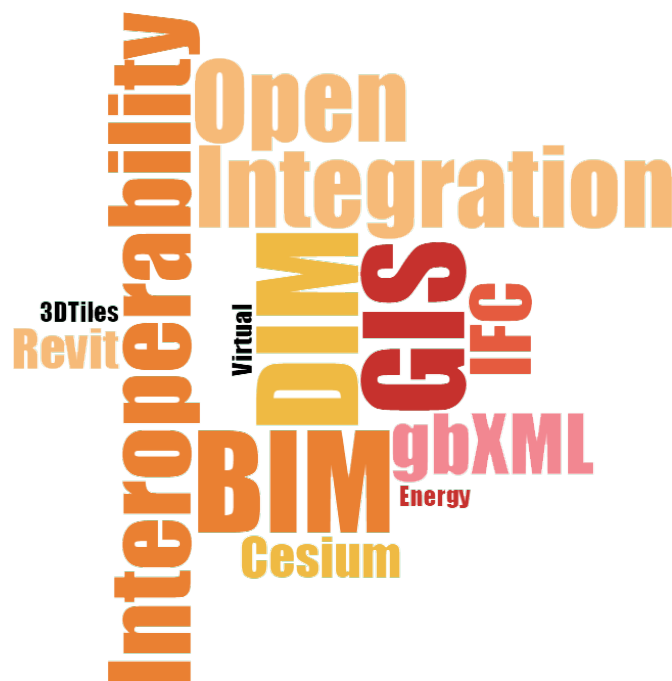
Ottobre 2017



# ABSTRACT

Recent years have seen an increasing number of technologies and spatial tools for enabling better decision-making in the urban environment. They had a wide diffusion because of the need for municipalities to be planned more efficiently, to accommodate growing populations while mitigating urban sprawl, making in this way a more advanced human and environment oriented planning.

At the same time the AEC has experienced a radical transformation, lead by the technological innovations brought by the raise and development of the Building Information Modeling (BIM) and the Geographic Information System (GIS) technologies. Both have imported in the world of architectural and territorial design powerful tools able to give great data richness to the projects. In this innovative context, this dissertation intend to find a connection between these two worlds, which are actually quite unable to communicate and share their immense fortune of knowledges. More specifically, it has been developed a methodology able to export geometries and informations from a BIM model, to convert them into a GIS compatible format and finally create a dashboard tool to collect these data, focusing on the definition of standardised procedures and on the intent of pursuing an "open" and "integrated" approach, by the use of open source specifications and softwares.







# TABLE OF CONTENTS

<b>1. INTRODUCTION</b>	<b>1</b>
1.1. BIM/GIS INTEROPERABILITY	2
1.2. VIRTUAL GLOBES	4
<b>2. PROJECT'S DESCRIPTION AND OBJECTIVES</b>	<b>7</b>
2.1. PROJECT DESCRIPTION	8
2.1.1. DESCRIPTION	8
2.1.2. CASE BUILDINGS	11
2.1.3. STATE OF THE ART	13
2.2. PROPOSED OBJECTIVES	14
2.2.1. THEORETICAL APPROACH: OPEN BIM AND INTEGRATED BIM	14
2.2.2. OBJECTIVES' DEFINITION	22
2.2.3. ATTENDED RESULTS	26
<b>3. PROPOSED METHODOLOGY</b>	<b>29</b>
3.1. GENERAL DESCRIPTION	30
3.2. PLANNING PHASE	32
3.3. BIM PHASE	37
3.3.1. PRELIMINARY OPERATIONS	37
3.3.2. RVT - GBXML EXPORTATION	45
3.3.3. RVT - IFC EXPORTATION	63
3.3.4. CONNECTIONS BETWEEN REVIT MODEL AND EXCHANGE MODELS	87
3.4. DIM PHASE	93
3.4.1. VISUALIZATION ON CESIUM	93
3.4.2. CONVERSION TO GLTF FORMAT	104
3.4.3. IMPORTATION AND VISUALIZATION	108
<b>4. RESULTS ANALYSIS</b>	<b>113</b>
4.1. VISUALIZATION	114
4.1.1. OBJECTS AND ATTRIBUTES REPRESENTATION FROM GBXML AND IFC MODELS	114
4.1.2. COMPARISON AND BEST CASE CHOICE	118
4.2. METHODOLOGY	121
<b>5. CONCLUSIONS</b>	<b>125</b>
5.1. FINAL ASSESSMENTS	126
5.2. FUTURE DEVELOPMENTS	127
<b>ANNEXES</b>	<b>139</b>
A. GBXML GEOMETRIES' REPRESENTATION	140



# IMAGES<sup>1</sup> INDEX

FIG. 2-1 - OPEN BIM LOGO.....	14
FIG. 2-2 - BIM CAPABILITY STAGES.....	17
FIG. 2-3 - NBIMS I-CMM CHART.....	18
FIG. 2-4 - BIM MATURITY LEVELS.....	20
FIG. 2-5 - BEW - RICHARD BIM MATURITY MODEL DIAGRAM .....	22
FIG. 2-6 - PROJECT'S STRUCTURE.....	23
FIG. 2-7 - EXPECTED OBJECTIVES.....	25
FIG. 2-8 - ATTENDED LOD1 VISUALIZATION.....	26
FIG. 2-9 - ATTENDED LOD2 VISUALIZATION.....	26
FIG. 2-10 - ATTENDED LOD3 VISUALIZATION (STEP A ON LEFT, STEP B ON RIGHT).....	26
FIG. 3-1 - DIAGRAM OF THE PROCEDURE.....	31
FIG. 3-2 - SENSORS <sup>1</sup> FAMILY EDITING.....	37
FIG. 3-3 - PROJECT TEMPLATE FILE.....	38
FIG. 3-4 - GEOREFERENCING (STEP 1).....	39
FIG. 3-5 - GEOREFERENCING (STEP 2).....	40
FIG. 3-6 - GEOREFERENCING (STEP 3).....	41
FIG. 3-7 - GEOREFERENCING (STEP 4).....	42
FIG. 3-8 - REAL NORTH ROTATION (STEP 1).....	42
FIG. 3-9 - REAL NORTH ROTATION (STEP 2).....	43
FIG. 3-10 - REAL NORTH ROTATION (STEP 3).....	43
FIG. 3-11 - SHARED PARAMETERS FILE.....	44
FIG. 3-12 - GBXML LOGO.....	45
FIG. 3-13 - GENERAL STRUCTURE OF THE GBXML MODEL.....	48
FIG. 3-14 - CAMPUS ELEMENT DEFINITION.....	51
FIG. 3-15 - CONSTRUCTION ELEMENT, LAYER ELEMENT AND MATERIAL ELE- MENT DEFINITION.....	53
FIG. 3-17 - SECTION VIEW AND GBXML 3D VIEW OF THE SENSOR ROOM.....	56
FIG. 3-16 - CREATION OF THE SENSOR ROOM'S CONTAINER.....	56
FIG. 3-18 - AREAS AND VOLUMES COMPUTING SETTING WINDOW.....	57
FIG. 3-19 - ENERGY SETTINGS WINDOW.....	58
FIG. 3-20 - GBXML EXPORT WINDOW (ROOM DETAIL).....	60
FIG. 3-21 - GBXML EXPORT WINDOW (ANALYTICAL SURFACES DETAIL).....	61
FIG. 3-22 - XML EDITOR INTERFACE VIEW.....	62
FIG. 3-23 - IFC LOGO.....	63
FIG. 3-24 - EXPRESS-G REPRESENTATION OF ENTITY DATATYPE.....	66
FIG. 3-25 - EXPRESS-G REPRESENTATION OF ENUMERATION DATATYPE.....	66
FIG. 3-26 - EXPRESS-G REPRESENTATION OF DEFINED DATATYPE.....	66
FIG. 3-27 - EXPRESS-G REPRESENTATION OF SELECT DATATYPE.....	66
FIG. 3-28 - EXPRESS-G REPRESENTATION OF SIMPLE DATATYPE.....	66
FIG. 3-30 - EXPRESS-G REPRESENTATION OF SUBTYPING.....	68
FIG. 3-29 - EXPRESS-G REPRESENTATION OF ATTRIBUTE.....	68
FIG. 3-32 - OBJECTS, PRODUCTS AND ELEMENTS DEFINITION.....	69
FIG. 3-31 - IFCROOT AND FUNDAMENTAL CLASSES.....	69
FIG. 3-33 - MANDATORY AND OPTIONAL (IN GREY) LEVELS OF THE SPATIAL STRUCTURE.....	70
FIG. 3-34 - DEFINITION OF IFCOBJECTPLACEMENT.....	72
FIG. 3-35 - DEFINITION OF IFCPRODUCTREPRESENTATION.....	73

FIG. 3-36 - DEFINITION OF IFCPROPERTY.....	74
FIG. 3-37 - PROPERTIES ATTACHMENT.....	76
FIG. 3-38 - REVIT'S BROWSER VIEW OF THE 3D IFC-DEDICATED VIEW.....	77
FIG. 3-39 - REVIT'S BROWSER VIEW OF THE LIST OF SCHEDULES.....	79
FIG. 3-41 - IFC MAPPING FILE.....	80
FIG. 3-40 - IFC EXPORT CLASSES DIALOG PANEL.....	80
FIG. 3-42 - IFC EXPORT MAIN PANEL.....	81
FIG. 3-43 - IFC EXPORT-SETUP PANEL, TAB 1.....	82
FIG. 3-44 - IFC EXPORT-SETUP PANEL, TAB 2.....	83
FIG. 3-45 - IFC EXPORT-SETUP PANEL, TAB 3.....	83
FIG. 3-46 - IFC EXPORT-ASSIGNMENTS PANEL.....	84
FIG. 3-47 - BIMVISION INTERFACE.....	85
FIG. 3-48 - IFCQUICKBROWSER INTERFACE.....	86
FIG. 3-49 - IFC-GUID BASE-64 CHARACTER ENCODING MAPPING.....	88
FIG. 3-50 - REVIT'S SELECT ELEMENTS BY ID WINDOW.....	89
FIG. 3-51 - REVIT'S ELEMENT ID CHECKING USING THE SELECT ELEMENTS BY ID COMMAND.....	90
FIG. 3-52 - IFCGUID CHECKING IN THE REVIT ELEMENT'S PROPERTY.....	90
FIG. 3-53 - IFCGUID AND ELEMENT ID CHECKING ON BIMVISION.....	91
FIG. 3-54 - OVERVIEW OF THE COMMUNICATION BETWEEN REVIT MODEL AND EXCHANGE MODELS.....	92
FIG. 3-55 - DIM ENVIRONMENT DEFINITION.....	93
FIG. 3-56 - CESIUM LOGO.....	93
FIG. 3-59 - SUBDIVISION OF THE SCENE IN TILES AND CHILD TILES.....	95
FIG. 3-58 - TILES' TREE STRUCTURE.....	95
FIG. 3-57 - 3DTILES LOGO.PNG.....	95
FIG. 3-60 - TILES' REPRESENTATION ON CESIUM.....	96
FIG. 3-61 - JSON'S MAIN ENTITIES DEFINITION.....	98
FIG. 3-62 - 3DTILES' FILES ORGANIZATION.....	99
FIG. 3-63 - GLTF LOGO.....	99
FIG. 3-64 - GLTF ASSET'S COMPOSITION.....	101
FIG. 3-65 - GLTF ASSET'S STRUCTURE.....	102
FIG. 3-66 - GLTF GEOMETRIES' REPRESENTATION.....	103
FIG. 3-67 - OVERALL CONVERSION PROCESS SCHEMA.....	104
FIG. 3-68 - FOLDER CONTATING THE TILESET.JSON AND THE B3DM FILES .....	108
FIG. 3-69 - DASHBOARD INTERFACE.....	110
FIG. 3-70 - DASHBOARD VISUALIZATION FROM SMARTPHONE.....	111
FIG. 4-1 - LOD VISUALIZATION OF THE GBXML BASED MODEL.....	114
FIG. 4-2 - GBXML VISUALIZATION ISSUES.....	115
FIG. 4-3 - LOD VISUALIZATION OF THE IFC BASED MODEL.....	115
FIG. 4-4 - IFC VISUALIZATION ISSUES.....	117
FIG. 4-5 - NOMENCLATURE DECODING.....	122

## TABLES' INDEX

TAB. 2-1 - "BY PLATFORM" AND "OPEN" APPROACHES DESCRIPTION.....	14
TAB. 2-2 - BIM CAPABILITY STAGES DESCRIPTION.....	16
TAB. 2-3 - BIMMI LEVELS DESCRIPTION.....	19
TAB. 2-4 - BEW - RICHARD'S LEVELS DESCRIPTION.....	21
TAB. 3-1 - LOD DEFINITION.....	33
TAB. 3-2 - CASE STUDIES' COMPARISON.....	36
TAB. 3-3 - KML AND GLTF COMPARISON.....	100



# 1. INTRODUCTION

## 1.1. BIM/GIS INTEROPERABILITY

The integration of data through geographic information system (GIS)-based building information modeling (BIM) has recently emerged as an important area of research. Several studies have investigated the benefits of the effective integration of BIM and GIS.

This process generally involves the extraction and transformation of information required by each stakeholder in the relevant project. GIS and BIM are similar in that they both model spatial information — the former is used for outdoor modeling and the latter for indoor modeling — and have common use cases, such as location-based municipal facilities information queries and management. In order to realize use cases based on BIM and GIS, effective interoperability between GIS and BIM should be supported by an appropriate platform. [1]

The integration of BIM and GIS can offer substantial benefits to manage the planning process during the design and construction phases. While BIM systems focus on developing objects with the maximum level of detail in geometry, GIS are applied to analyze the objects, which already exist in the physical environment, in a most abstract way. The major difficulty in integrating BIM and GIS systems reflects their incompatibility such as the modeling environment and reference system (e.g., GIS data are georeferenced and usually two dimensional while the BIM data are three-dimensional objects located within local coordinate systems). Although these two technologies have evolved from distinctly different beginnings, both can benefit from each other if they could exchange data effectively. As BIM technology is mainly centered on indoor environments, GIS can extend the benefits and applicability of existing building models to the outdoor environment.

However, it is not an easy task to transfer data from BIM to GIS or vice versa without consideration of data format and meaning. Current state-of-the-art BIM (or GIS) tools enable the data exchange between the systems by using a common data format. Therefore, the users are able to access data from a different software program and exchange data within the BIM (or GIS) domain. However, it requires the user to have a thorough understanding of both systems and their functionalities. The integration tools and current standards lack the ability to help the user to convey meaning, which is interpretable by both construction project participants as well as BIM and GIS tools. In order to fully integrate GIS and BIM, there is a need to provide interoperability at the semantic level.

The current approach to exchange and share building data between BIM applications is based on the exchange of industry foundation classes (IFC) files. While this approach was, and still remains, an effective way to hold and exchange data among various participants in a building, construction, or facility



management project, it does not provide semantic-based representation of knowledge for another domain (e.g., geospatial domain), and thus limits the capability of inferring additional knowledge. [2]

## 1.2. VIRTUAL GLOBES

The virtual globes can be categorized into two groups: the first, running on desktop devices and the second, running on web browsers, evolved in the last few years, thus the modern ones. Regarding the modern virtual globes, the two most outstanding examples in open source domain are NASA Web World Wind and Cesium. These two leverage the latest technologies for Web: JavaScript, WebGL and HTML5. They can be used by non-experts thanks to their API (Application Programming Interface) and customized conveniently to meet the needs of case studies. This means that almost any computer programmer without advanced programming skills can create their own virtual globe application and share it with everyone via Web. Moreover, they can run on any device (desktop or mobile) and major operating systems, without having to install additional software, such as plugins or extensions. As a result, modern and open source virtual globe APIs increase the opportunity to create three-dimensional maps and to let users interact with them. [6]

They can be divided also in another way: Closed source platforms are complete software packages targeted at end users who simply need to add data to create visualisation applications for particular areas or themes. Users often have to pay for closed source systems, although free versions with limited functionality are sometimes available. For closed source systems, the ownership of the software stays with the producer of the product which means the user cannot sell, distribute, copy and/or change the content of the software. As the owner has complete control, it puts the user at risk if for instance the owner decides to suddenly decommission the product. Open source platforms are often not as straight forward to use, so tend to be targeted at experienced users/developers who can support customisation (via coding). Open source platforms can be updated more quickly than closed source platforms when the user community gets enthusiastic. They may be more reliable for long term projects due to the broader base of developers which reduces the likelihood that they will disappear if the original creator stops working on them. [35]

These systems are revolutionizing earth observation data access and integration in two primary ways:

- **Democratization of access.** The popularity of the openly accessible virtual globes extends far beyond the traditional professional communities engaged in geospatial science and commerce. The number of people interactively viewing and extracting content from earth observations such as satellite imagery is on a rapid upward swing as a result of virtual globes. For the technical users virtual globes have vastly reduced the overhead associated with accessing global archives of satellite imagery by

eliminating purchase costs and effort required to stage and manage large image holdings.

- **Democratization of content contribution.** Users are able to make links to their own earth observation data via web mapping services and insert site specific content (such as descriptions and photographs) which can be openly accessed by the broader community. This has made it possible to integrate earth observation data from diverse sources, enable increased productivity for individual projects and studies. Virtual globes are presently used extensively in areas including education, research/ collaboration, and disaster response. The current wealth of technologies, expanding bandwidth, changing user expectations, and data available via the geospatial web are driving the rapid development of virtual globes. This paper presents a brief overview of virtual globes over the last decade, reviews the current capabilities and applications for virtual globes, and envisions what may be anticipated in the coming years. [7]

As cities become more densely populated there is a need for more advanced planning tools to assist decision-makers, planners and communities to collectively plan for more sustainable, productive and resilient cities. The recent emergence of the concept of geodesign, which embraces the intersection of geography and design, has seen the development of a new suite of planning support tools to aid the design and planning of cities, particularly at the scale of precincts. [3]



## 2. PROJECT'S DESCRIPTION AND OBJECTIVES

## 2.1. PROJECT DESCRIPTION

### 2.1.1. DESCRIPTION

Before introducing the specific objectives and targets of the present research, it's fundamental to describe the context in which it will be inserted and the generic aspect involved.

#### BIM/GIS interoperability

Referring to the previous chapter, this project is intended to pursue the process of integration between the BIM and the GIS universes, bringing to the same platform data and geometries from these two IT environments.

This process involves a sequence of conversion and transferring procedures intended to maintain and finally show informations and scenes from projects previously designed on sectoral and non communicating BIM and GIS softwares. This project wants to focus on the BIM side, where the main obstacle to face is the impossibility of using directly the native format of BIM softwares (as RVT is for Revit) as exchange format: this means we have to find alternative ways in order to export data not only to a BIM exclusive use, but for universal and generic end practices.

Fortunately, the BIM softwares, like Revit, allows different exportation possibilities, some maybe focused only on specific use (like gbXML for energy analysis), other more suitable for general purposes (like IFC), and this give us the possibility of finding a indirect solution with the available means.

#### Virtual globe as means of dissemination

As mentioned in the previous chapter, virtual globes are expanding very fast in the daily use and they are getting more and more applications. One of these is surely the implementation with realistic 3D building models, intended to represents whole cities around the world for different purposes. Since today, this use of the virtual globes has shown less or more elaborated buildings' representations, where several systems of modeling and photogrammetry have given in last years successful results in terms of buildings' representation.

The second aspect which characterizes the late development of the virtual globes' market is the visualization of geographic and demographic data, as true GIS softwares, where with, interactive systems of stylization and visualization, we are able to benefit from information from different disciplines, directly applied to virtual objects and environments. Over the past several years virtual globes have revolutionizing access to earth observation data and integration. They have lowered the start-up threshold for access to global satellite data and opened up new possibilities for collaborative research and product generation [7].

In this optic, the new frontier of virtual globes technologies, which this project intends to reach, is the implementation on a web based virtual globe of informations from a BIM based modeling made on district scale, for the creation of a DIM interface, described as in "1.1. BIM/GIS interoperability".

These intents have been faced in two projects, described briefly below, that inspired the idea of this essay.

District Information  
Modeling

## 2.1.1.1. DIMMER AND EEB INHERITANCE

### 2.1.1.1.1. DIMMER PROJECT

This research takes its origins with the development of the DIMMER ("District Information Modelling and Management for Energy Reduction") project, a three years program started on 1 October 2013 which has involved several companies from all around Europe.

Description

DIMMER was a project intended "to integrate BIM and district level 3D models with real-time data from sensors and user feedback to analyse and correlate buildings utilization and provide real-time feedback about energy-related behaviours" [44].

This project has involved two cities' districts, in Turin and Manchester, considering a total of 15 buildings (7 in Turin, 9 in Manchester), both public (university campuses, schools) and private, of different period of construction.

This project aimed to unlock potentiality of BIM technologies, applying them to a district level (DIM) and combining them with the ICT (Information and Communication Technologies) on a GIS interface. In this project, progresses in real-time monitoring and BIM have been combined to create a District Information Model and Management system, a simulation framework for monitoring district level energy usage, aimed at a consistent reduction in terms of emissions and consumptions and at enabling a more efficient energy distribution policies.

Informations, collected from sensors installed in the buildings and along the distribution networks and from users' personal devices, were gathered in a dedicated middleware and then represented on end-user applications.

An important role for this project was in fact dedicated to the development of a dashboard for dynamic monitoring and management of energy consumption, visualising heating data in case of Turin pilot and district heating, HVAC and electricity data for Manchester pilot.

Dashboard

Description of the major functionalities implemented into the Dashboard [8]:

- a) Geospatial capabilities and functionality for visualisation and for analysis of geospatial data patterns;

- b) Authentication and user profiling, possibility for definition of specific user with associated roles;
- c) Visualisation of data through 2D and 3D maps;
- d) Analysis of the information related to the pilot buildings and district as the whole (construction period, volume, heating system, temperature, floors number, address, name of building, building category);
- e) Energy consumption analysis through tables and charts, e.g. energy consumption, energy power, energy indicators such as thermal behaviour index;
- f) The dashboard is accessible through a specific user profile, depending on the target user; each target user can access specific data/information, which are not available for everyone.

gbXML as format for simulations and GIS 3D representation

In particular, for what involved the transmission of the BIM models for the DIM visualization, the strategy applied was that of sharing a gbXML model of the building. This choice had many advantages, including:

- gbXML model had to be in any way considered for the energy analysis tools. So we could use the same model both for simulation both for GIS visualization;
- gbXML, in terms of visualization, is very light, as it is composed only by analytical surfaces.

However, many issues, reported in chapter "4. Results analysis", dug up in terms of manageability of this format for the use it was intended for, and so we had to find alternatives.

#### 2.1.1.1.2. EEB PROJECT

This project was born as a continuation of the DIMMER project, intended to apply the same strategy to the city of Settimo Torinese, near Turin.

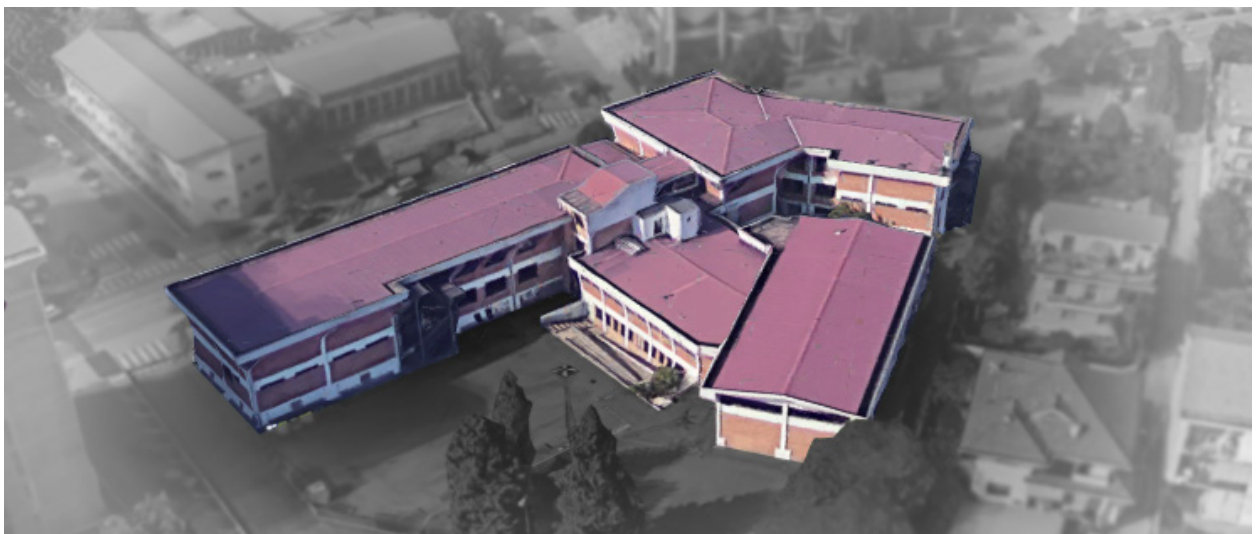
The EEB (Zero Energy Buildings in Smart Urban Districts) is intended "to pursue the increase the energy efficiency in buildings, and then of urban districts, through the pervasive use of technologies for the real-time monitoring and control of environmental parameters and of the energy production/consumption by means of smart devices" [45].



### 2.1.2. CASE BUILDINGS

Since the present research is intended to pursue this interoperability between BIM and GIS technologies, it was useful to apply it to an active project like EeB, where it could be possible to have access to sensors data and to make a contribution to its development.

For this reason, we have considered the three buildings involved in the project, that are presented below.





### 2.1.3. STATE OF THE ART

Referring to the EeB project, which contains the most recent step undertaken in the direction which this project is intended to, we can assess the actual state of the art, analysing the used procedures and the results obtained.

## 2.2. PROPOSED OBJECTIVES

### 2.2.1. THEORETICAL APPROACH: OPEN BIM AND INTEGRATED BIM

#### 2.2.1.1. OPEN BIM

The first target of the project is to make it fit with the Open BIM program, in the light of the numerous advantages this will add.

##### Open BIM Certification



Fig. 2-1 - Open BIM logo

Open BIM is a marketing campaign started by two important software houses like Tekla and Graphisoft, aimed at encouraging the promotion and the coordinate development of the Open BIM Certification, which is a technical certification system intended to help AEC software houses to improve, test and certify that the data produced by their softwares communicate correctly with the other Open BIM applications.

##### Open BIM approach

But the most important objective of the Open BIM program is the definition of a "open" design approach, which separates clearly from the previous "by platform", in particular for its benefits in terms of collaboration. The "by platform" collaboration is based on different application from the same software developer, while the "open" approach allows a collaboration based on different software solutions.

Tab. 2-1 - "By platform" and "Open" approaches description

BY PLATFORM APPROACH (OLD)	OPEN APPROACH (NEW)
NO data conversion	Transparent workflow
Limited external data usage	System independent
Compatibility issues	Overall BIM compatibility

In other words, Open BIM defines a transparent and open workflow, allowing the several stakeholders involved during all the building's lifecycle to collaborate independently of the softwares used, and at the same time it defines a common language, based on convertible and shared data and easily applicable to large scale projects.

## 2.2.1.2. INTEGRATED BIM

The second main intent of this project is to develop a project's strategy which could be assimilated within the concept of integrated BIM.

The definition of integrated BIM derives from the debates about the definition of capability and maturity level of the BIM, started in the early '00s when this new technology entered the world of constructions.

In spite of an increasing diffusion of the BIM technologies and the abundance of technical discussions and literature professing the capability of BIM methodologies to increase productivity and efficiency, BIM has not found yet the availability of metrics and tools to measure this benefits. "This mismatch between expected BIM deliverables and unforeseen BIM requirements increases the risks, costs and difficulties associated with BIM implementation, allows the proliferation of 'BIM wash' – falsely professing the ability to deliver BIM services or products - and prevents industry players from achieving their BIM potential" [9].

One of the greatest obstacles for the global adoption of the BIM technologies is the lack of standard productivity data which could estimate BIM's financial sustainability. This involve us in the "chicken - egg" causality dilemma: BIM should be practiced for its economical benefits, but without knowing exactly these benefits we are reluctant to adopt it [13].

There is, in short, the need of a BIM performance metrics, which could help teams and organizations to evaluate their own BIM competencies or compare them with an industry benchmark.

The first solution to this set of problems is to define a BIM framework or, in other words, to organize the domain knowledge around the BIM technology.

BIM Framework

This operation will allow practitioners and educators to gain advantage by a structured subdivision of the domain knowledge, which can promote understanding and technical development by presetting data and arguments in ordered sections [11].

Succar [9] indicates five components which defines the BIM domain and enable accurate BIM performance measurement:

- BIM Capability stages;
- BIM Maturity levels;
- BIM Competency sets;
- BIM Organisational scales;
- BIM Granularity levels.

Focusing on the two first components, we can initially give a proper definition of

"capability" and "maturity".

The capability can be defined as "the basic ability to perform a task or deliver a BIM service" [9] of a team or organization which works using BIM services. The capability can be described through three BIM Capability Stages, that defines the minimum BIM requirements that those teams or organizations must reach. These stages, as represented in Fig. 2–2, lead an evolution from a pre- BIM, the status point before the BIM implementation, and the post- BIM environment, a non well defined ending point representing the final evolving target of employing virtually integrated tools and concepts.

These three stages can be described as it follows.

Tab. 2–2 - BIM Capability Stages description

<p><b>BIM STAGE 1</b></p> 	<p><b>OBJECT - BASED MODELING</b></p>	<ul style="list-style-type: none"> <li>• Object-based 3D parametric software tool</li> <li>• Single-disciplinary models</li> <li>• No modifiable parametric attributes</li> <li>• Data exchanges between project stakeholders are unidirectional and communications continue to be synchronous and disjointed</li> </ul>
<p><b>BIM STAGE 2</b></p> 	<p><b>MODEL - BASED COLLABORATION</b></p>	<ul style="list-style-type: none"> <li>• Players actively collaborate with other disciplinary players</li> <li>• Communications between BIM players continue to be asynchronous</li> <li>• Higher detail construction models move forward and replace (partially or fully) lower-detail design models</li> </ul>
<p><b>BIM STAGE 3</b></p> 	<p><b>NETWORK - BASED INTEGRATION</b></p>	<ul style="list-style-type: none"> <li>• Semantically-rich integrated models</li> <li>• Integration can be achieved through model server technologies</li> <li>• Models become interdisciplinary nD models</li> <li>• Synchronous interchange of model and document-based data</li> </ul>

Between each of these stages we can identify four theoretical steps (step A, step B, step C and step D) which indicate the incremental working from a stage to another one, representing so the wide interval between the different stages along the continuum represent in the image below.

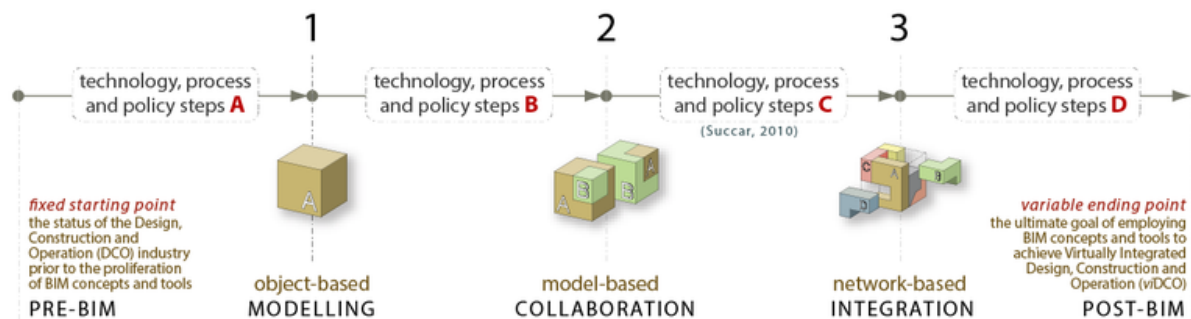


Fig. 2-2 - BIM Capability Stages

The maturity refers instead to "the quality, repeatability and degree of excellence within a BIM Capability" [9]. In other words, BIM Maturity is the ability to excel in performing a task or delivering a BIM service/ product.

BIM maturity

For the moment we can specify that, in general, the progression from a lower to a more advanced maturity level is characterised by the following aspects:

- Better control through performance targets and effective results;
- Better predictability by lowering in terms of cost, performance and competency variables;
- Greater effectiveness in reaching the defined goals, allowing to set new more ambitious ones.

At this point we bump into the problem of measuring these qualities and abilities which characterise the BIM maturity and defining maturity levels.

BIM Performance measurement

To address this issue, since BIM is included in the more generic IT universe, it has been analysed in its maturity aspects using Maturity Models and Tools developed for generic softwares and application fields but applicable to BIM and the AEC industry, as a basis for the development of BIM specific Maturity Model.

These models have been analysed in their suitability for the BIM application and for the definition of the different level of maturity and finally the BIM Maturity index.

The first effort to create a BIM specific performance model is attributed to the U.S National Building Information Modeling Standard (NBIMS) in 2007 with the name of Interactive Capability Maturity Model (I-CMM).

Interactive Capability Maturity Model

This model consists of 10 levels of maturity, each characterised by 11 parameters.



Maturity Level	A Data Richness	B Life-cycle Views	C Roles Or Disciplines	G Change Management	D Business Process	F Timeliness/ Response	E Delivery Method	H Graphical Information	I Spatial Capability	J Information Accuracy	K Interoperability/ IFC Support
1	Basic Core Data	No Complete Project Phase	No Single Role Fully Supported	No CM Capability	Separate Processes Not Integrated	Most Response Info manually re-collected - Slow	Single Point Access No IA	Primarily Text - No Technical Graphics	Not Spatially Located	No Ground Truth	No Interoperability
2	Expanded Data Set	Planning & Design	Only One Role Supported	Aware of CM	Few Bus Processes Collect Info	Most Response Info manually re-collected	Single Point Access w/ Limited IA	2D Non-Intelligent As Designed	Basic Spatial Location	Initial Ground Truth	Forced Interoperability
3	Enhanced Data Set	Add Construction/ Supply	Two Roles Partially Supported	Aware of CM and Root Cause Analysis	Some Bus Process Collect Info	Data Calls Not In BIM But Most Other Data Is	Network Access w/ Basic IA	NCS 2D Non-Intelligent As Designed	Spatially Located	Limited Ground Truth - Int Spaces	Limited Interoperability
4	Data Plus Some Information	Includes Construction/ Supply	Two Roles Fully Supported	Aware CM, RCA and Feedback	Most Bus Processes Collect Info	Limited Response Info Available In BIM	Network Access w/ Full IA	NCS 2D Intelligent As Designed	Located w/ Limited Info Sharing	Full Ground Truth - Int Spaces	Limited Info Transfers Between COTS
5	Data Plus Expanded Information	Includes Constr/Supply & Fabrication	Partial Plan, Design&Constr Supported	Implementing CM	All Business Process(BP) Collect Info	Most Response Info Available In BIM	Limited Web Enabled Services	NCS 2D Intelligent As-Built	Spatially located w/Metadata	Limited Ground Truth - Int & Ext	Most Info Transfers Between COTS
6	Data w/Limited Authoritative Information	Add Limited Operations & Warranty	Plan, Design & Construction Supported	Initial CM process implemented	Few BP Collect & Maintain Info	All Response Info Available In BIM	Full Web Enabled Services	NCS 2D Intelligent And Current	Spatially located w/Full Info Share	Full Ground Truth - Int And Ext	Full Info Transfers Between COTS
7	Data w/ Mostly Authoritative Information	Includes Operations & Warranty	Partial Ops & Sustainment Supported	CM process in place and early implementation of root cause analysis	Some BP Collect & Maintain Info	All Response Info From BIM & Timely	Full Web Enabled Services w/IA	3D - Intelligent Graphics	Part of a limited GIS	Limited Comp Areas & Ground Truth	Limited Info Uses IFC's For Interoperability
8	Completely Authoritative Information	Add Financial	Operations & Sustainment Supported	CM and RCA capability implemented and being used	All BP Collect & Maintain Info	Limited Real Time Access From BIM	Web Enabled Services - Secure	3D - Current And Intelligent	Part of a more complete GIS	Full Computed Areas & Ground Truth	Expanded Info Uses IFC's For Interoperability
9	Limited Knowledge Management	Full Facility Life-cycle Collection	All Facility Life-Cycle Roles Supported	Business processes are sustained by CM using RCA and Feedback loops	Some BP Collect&Maint In Real Time	Full Real Time Access From BIM	Netcentric SOA Based CAC Access	4D - Add Time	Integrated into a complete GIS	Comp GT w/Limited Metrics	Most Info Uses IFC's For Interoperability
10	Full Knowledge Management	Supports External Efforts	Internal and External Roles Supported	Business processes are routinely sustained by CM, RCA and Feedback loops	All BP Collect&Maint In Real Time	Real Time Access w/ Live Feeds	Netcentric SOA Role Based CAC	nD - Time & Cost	Integrated into GIS w/ Full Info Flow	Computed Ground Truth w/Full Metrics	All Info Uses IFC's For Interoperability

© NIBS 2007

Fig. 2-3 - NBIMS I-CMM Chart

However, in its current form, the I-CMM tool suffer from structural limitations that may restrict its usefulness and usability.

#### Building Information Modelling Maturity Index

In 2010, Succar developed his own Building Information Modelling Maturity Index (BIMMI), characterized by five maturity levels, whose names took reference from the other generic IT Maturity Models mentioned above. These levels are listed in the chart below.



Tab. 2–3 - BIMMI levels description

<p><b>LEVEL A: INITIAL</b></p>	<ul style="list-style-type: none"> <li>• Absence of an overall strategy</li> <li>• Shortage of defined processes and policies</li> <li>• BIM adoption is partially achieved</li> <li>• Collaboration capabilities incompatible with those of project partners</li> <li>• No pre-defined process guides, standards or interchange protocols</li> </ul>
<p><b>LEVEL B: DEFINED</b></p>	<ul style="list-style-type: none"> <li>• Business opportunities arising from BIM are identified but not yet exploited</li> <li>• Basic BIM guidelines are available</li> <li>• Collaboration with project partners starts to be organized</li> <li>• Primitive predefined process guides, standards and interchange protocols</li> <li>• Responsibilities are distributed through contractual means</li> </ul>
<p><b>LEVEL C: MANAGED</b></p>	<ul style="list-style-type: none"> <li>• The vision to implement BIM is communicated and understood by most staff</li> <li>• BIM implementation strategy is detailed</li> <li>• Business opportunities coming from BIM are acknowledged</li> <li>• BIM roles are institutionalized</li> <li>• Detailed standards and quality plans</li> <li>• Collaboration responsibilities, risks and rewards are clear</li> </ul>
<p><b>LEVEL D: INTEGRATED</b></p>	<ul style="list-style-type: none"> <li>• Software selection and deployment follows strategic objectives</li> <li>• Modelling deliverables are well synchronized across project</li> <li>• Knowledge is integrated into easy accessible organizational systems</li> <li>• BIM roles and competency targets are imbedded within the organization</li> <li>• BIM standards and performance benchmarks are incorporated</li> </ul>

## LEVEL E: OPTIMIZED

- BIM implementation strategy and its effects on organizational models are continuously revisited and realigned with other strategies
- Alterations to processes or policies are proactively implemented
- Selection/use of software tools is continuously revisited to enhance productivity and align with strategic objectives
- Collaborative responsibilities, risks and rewards are continuously revisited and realigned
- Benchmarks are repetitively revisited

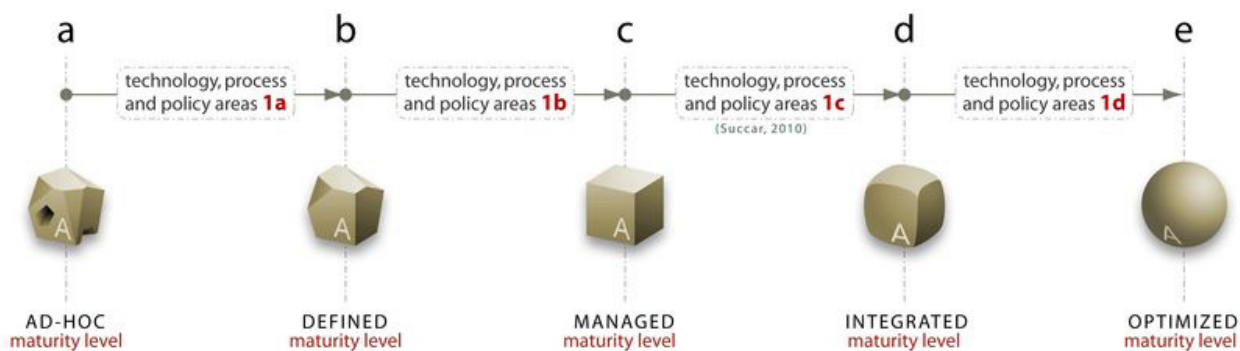


Fig. 2-4 - BIM Maturity levels

### Bew-Richard BIM Maturity Model

The following paragraph is dedicated to a UK specific BIM context, defined by the PAS 1192 regulation, which represents the main reference for the appliance of the BIM methodology in the UK territory.

PAS 1192-2, in particular, focuses on project delivery and it is referred to organizations and individuals responsible for the whole lifecycle of buildings and infrastructure [14].

One of the most interesting aspects of this regulation is the definition of a The UK maturity model was developed by Mark Bew and Mervyn Richards in 2008 [40]. There are many versions of the base model with subtle but meaningful differences; in this case its represented and analysed the 2016 version.

It is commonly known as the "BIM Wedge" because of its particular graphical shape (see Fig. 2-5). From the left to right, this chart is divided into four sections representing the increasing levels of maturity, listed in Tab. 2-4. These levels can be grouped into three categories, representing their state of definition:

- Stable (levels 0 and 1);
- Stabilising (level 2)
- Yet to stabilize (level 3)

In the lower part they're reported all the legislations which defines each level's

standard.

Tab. 2–4 - Bew- Richard's levels description

<b>LEVEL 0 BIM</b>	<ul style="list-style-type: none"> <li>• Effectively means no collaboration</li> <li>• 2D CAD drafting only is utilised, mainly for Production Information</li> <li>• Output and distribution is via paper or electronic prints, or a mixture of both</li> </ul>
<b>LEVEL 1 BIM</b>	<ul style="list-style-type: none"> <li>• Mixture of 3D CAD for concept work and 2D for approval documentation and Production Information</li> <li>• Electronic sharing of data is carried out from a common data environment (CDE)</li> <li>• Models are not shared between project team members</li> </ul>
<b>LEVEL 2 BIM</b>	<ul style="list-style-type: none"> <li>• Collaborative working, but not necessarily working on a single, shared model</li> <li>• Design information is shared through a common file format</li> </ul>
<b>LEVEL 3 BIM</b>	<ul style="list-style-type: none"> <li>• Full collaboration between all disciplines</li> <li>• Single, shared project model, held in a centralized repository</li> <li>• This is known as "Open BIM"</li> </ul>

The importance of these levels is certified by the fact that UK government has adopted these definitions in its Construction Strategy, forcing, for example, that all publicly-funded construction work must be undertaken by using Building Information Modelling to Level 2, by 2016 [16].

The main flaw of the Bew- Richards Maturity Model is that actually it's not a maturity model, in the true sense of the word, but it can be described more properly as a strategy model, a policy model or an industry roadmap [40]. This erroneous definition has several potential consequences, first of all the risk of confusing the strategic targets with compliance milestones with set standards and protocols.

Applied more widely outside the UK territory, the BIM Wedge loses much of its efficacy, but it can still be used for the definition of long-term objectives, since it shows itself like an open-ended model that invites to imagine subsequent levels and to add layers of knowledges on top of those already defined.

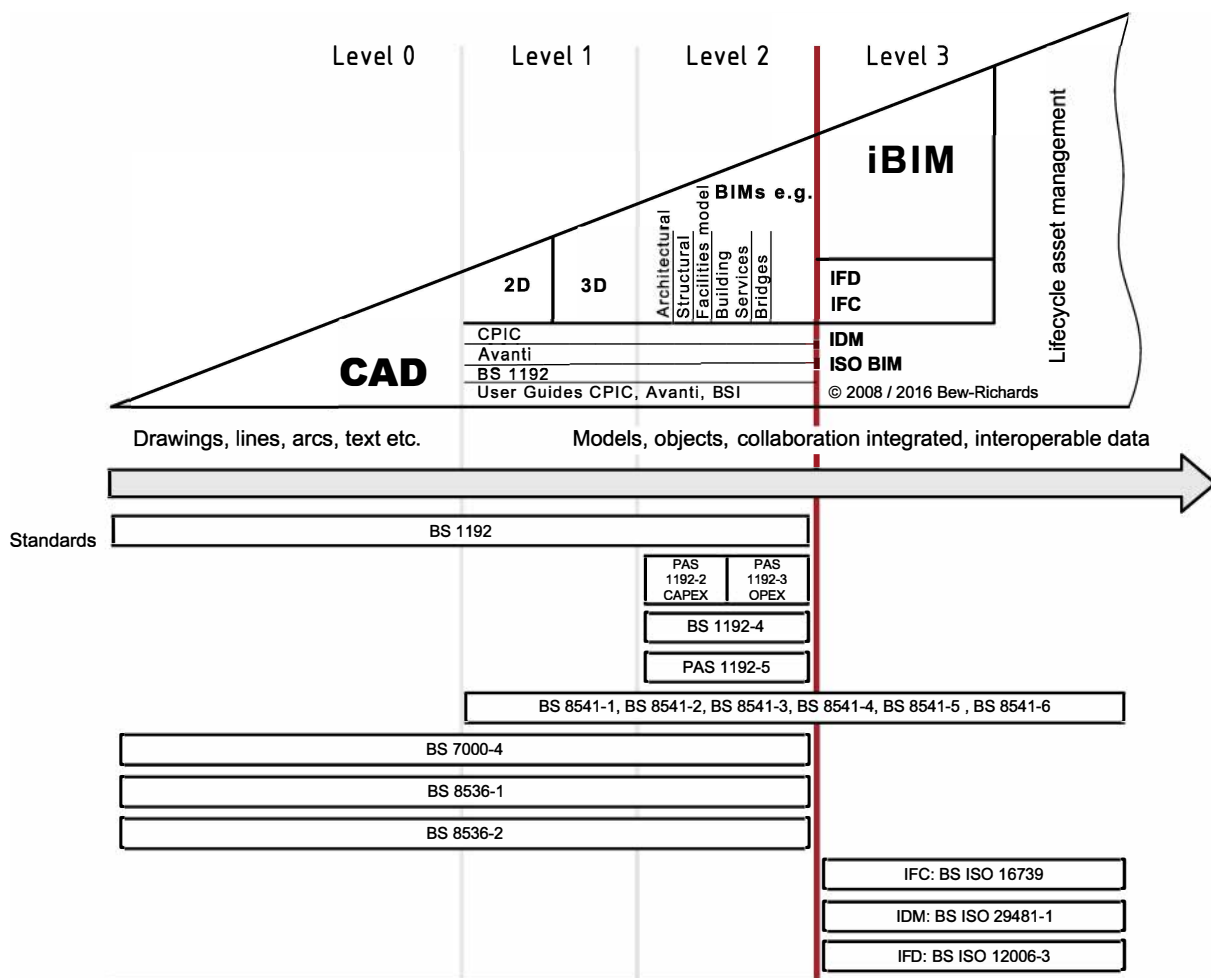


Fig. 2-5 - Bew- Richard BIM Maturity Model diagram

## 2.2.2. OBJECTIVES' DEFINITION

In view of this, we can start defining the milestones and the targets of the project. Bearing in mind the concepts of "open" and "integrated" BIM, it is possible to plan a collaborative, multidisciplinary and multiuser workflow that allows the development of open, coordinated and simultaneous projects between the various technicians or agents in an interactive way, through the progressive resolution of its various aspects.

Before planning a well defined methodology, it may be useful to create a conceptual map of the project in which we can highlight the characteristics of the project responding the above-mentioned notions of "open" and "integrated".

The concept of integrated BIM, in view of what explained above, is based on the synchronous interchange of interdisciplinary models and document-based data. For this reason it's necessary to elaborate a project with a network based structure, where data from different disciplines converge to the same storage environment, based on advanced server technologies, creating in this way what we call a "middleware".

The middleware is a connection software that consists of a set of services and application development environments that allow multiple entities (processes, objects, etc.), resident on one or more computers, to interact through an interconnection network in spite of differences in communication protocols, local system architectures, operating systems. In this way data from local platform are simultaneously shared and analysed and managed for a specific output service. In this case the output is limited to the conversion and loading of BIM models into a GIS environment.

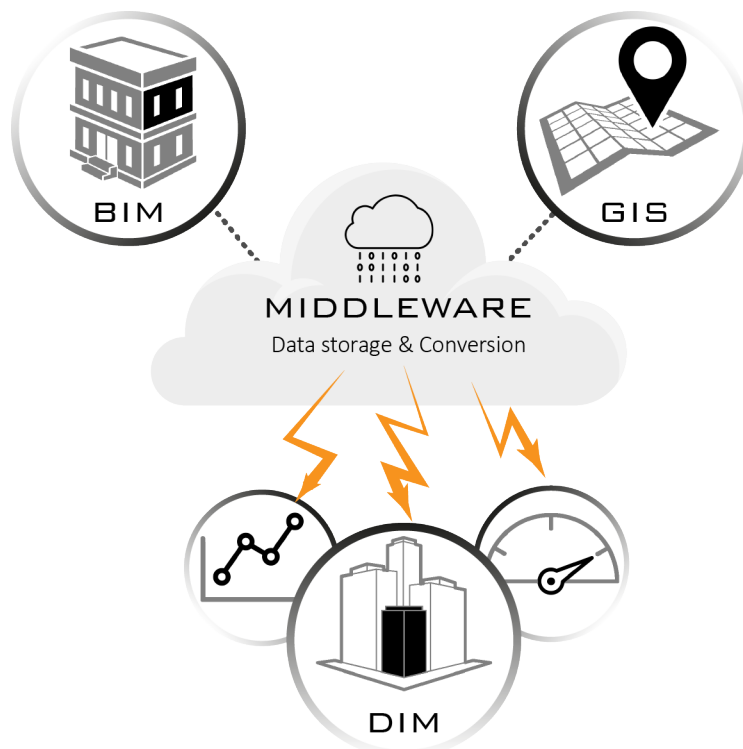


Fig. 2-6 - Project's structure

Considering that this research is marked on the implementation of BIM in district data visualization, GIS data won't be discussed in depth since its integration in the DIM environment is already resolved.

Focusing on the exportation and conversion operations analysed in the following chapter, we can highlight some aspects to consider along these procedures in order to give to the processes an "open" and "integrated" conformity:

- **Unique BIM model:** This is the starting point of the project. This choice is made with the purpose of avoiding the creation of independent BIM models for specific use only. We don't want a BIM model optimized for the structural analysis or energetic simulations. The objective is to maintain the uniqueness of the BIM model, but finding a way to optimize it for the different purposes, working for example on the creation of specific use model's views, like in the case of the IFC exportation.

- **Use of open standard exchange formats:** An open format (like IFC, gbXML and glTF) is a file format for storing digital data, defined by a published specification, usually maintained by a standard organization, and which can be used and implemented by anyone. For example, an open format can be implemented by both proprietary and free and open-source software, using the typical software licenses used by each. The result is that the project is not constrained to any specific application or program.
- **Use of a open source converter software:** Furthermore, applications used across the project must be "open source", which means softwares whose authors (more precisely, rights holders) make public the source code, promoting the free study and allowing independent programmers to make changes and extensions. For this reason, even if in the market it's possible to find sophisticated tools, it took an important part in the project the development of a project's specific software, used to convert the BIM models in a GIS ready format, responding to the definition of open source software.
- **Use of a web based and open source web globe:** This aspect, whose importance is restricted to the final output of the project, give us unlimited possibilities in terms of visualization and navigation. In particular, the use of this kind of web globe gives the opportunity to have free access to source codes, and then to a great customization of the product, and to apply it to any browser, device or system.
- **Creation of standard layouts and libraries:** This strategy gives the project a solid structure, where user's choices are constrained by pre defined lists of objects and attributes. This imposition is intended to limit the proliferation of random data and elements which can cause misunderstandings and redundancy errors. In this view, for example, also the specification of a well defined nomenclature can give an important contribute.
- **Automation of the processes:** Finally, another important aim of the project is to make the processes as automatic as possible. This will avoid the user to do customized operations, obtaining on the other hand a better control over the workflow, since every step made is tied to strong and pre defined rails. In facts, this is possible for example with the creation of fixed export set up which can be exported and applied to different BIM models.

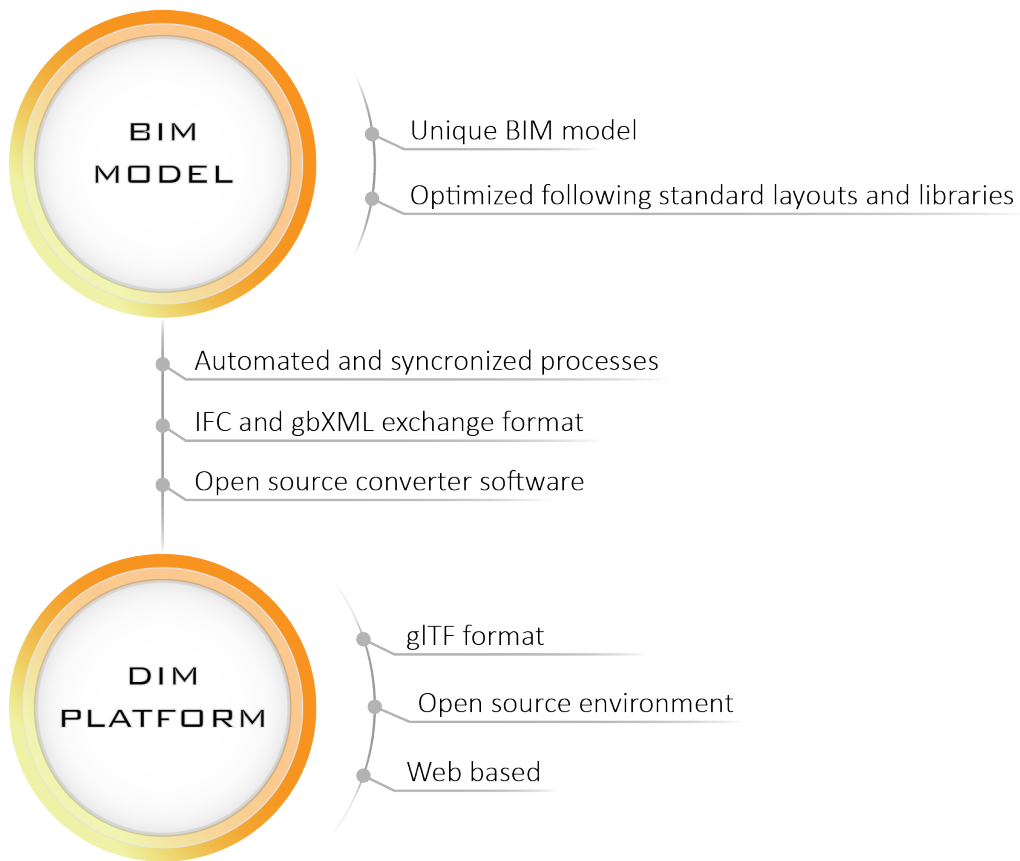


Fig. 2-7 - Expected objectives

## 2.2.3. ATTENDED RESULTS

Suddenly are exposed some concepts of the expected results in terms of visualization and navigation through the dashboard, divided per LoD.

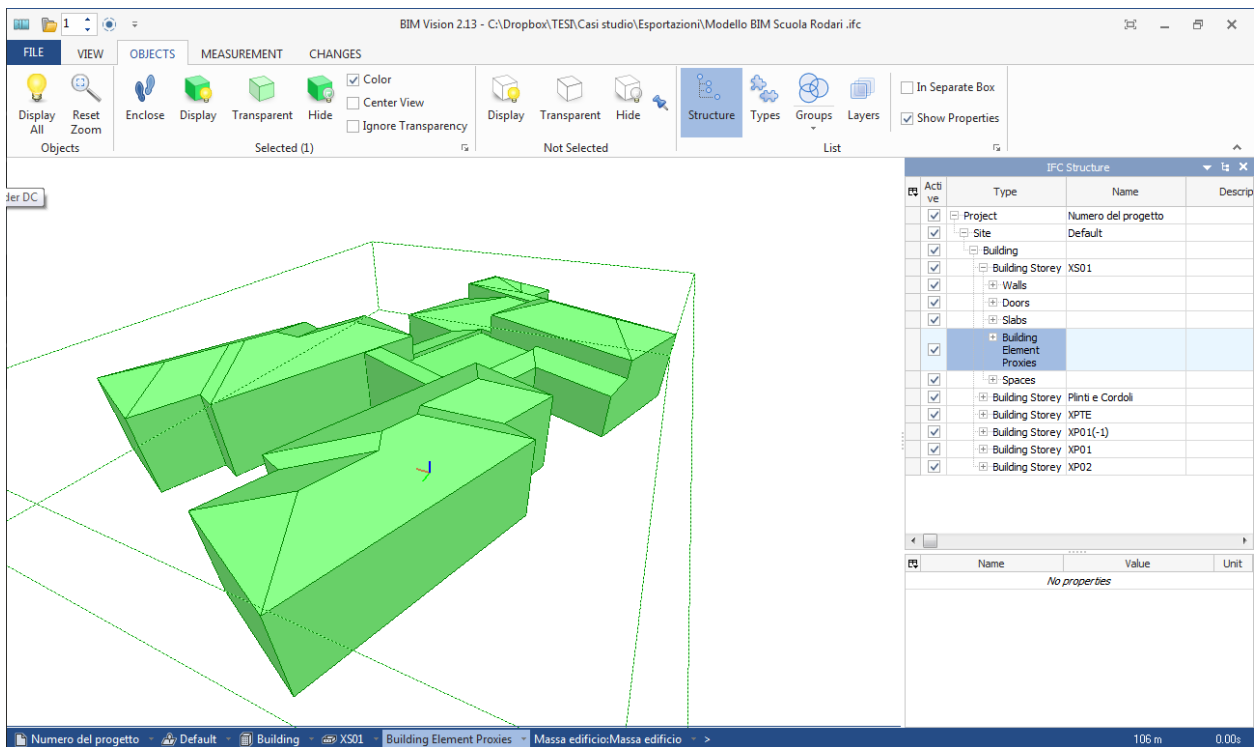


Fig. 2-8 - Attended LoD1 visualization

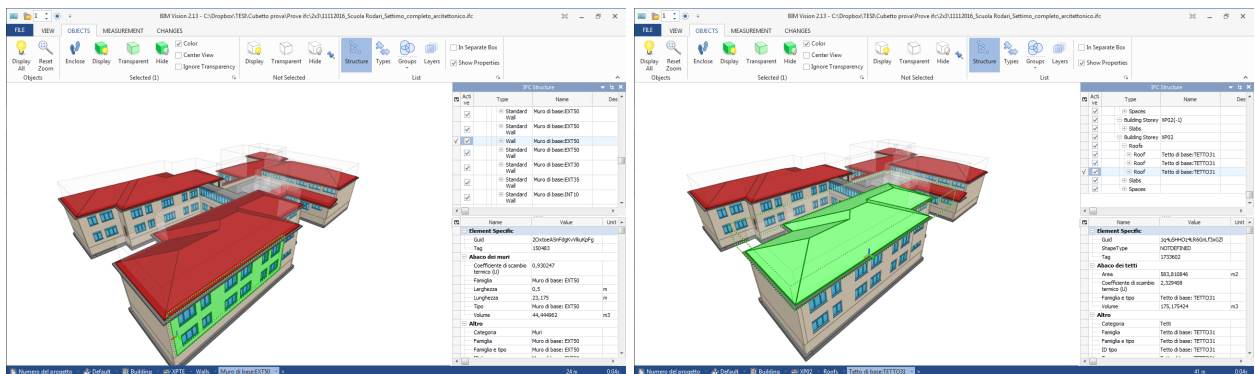


Fig. 2-9 - Attended LoD2 visualization

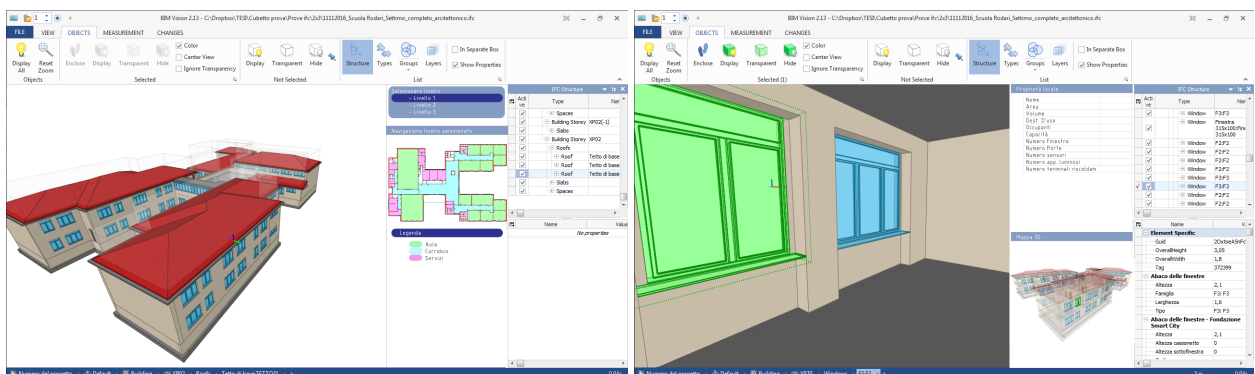


Fig. 2-10 - Attended LoD3 visualization (step A on left, step B on right)



The main interface is derived by the BIMVision software, in order to simulate the GIS environment we want to build. It is composed of a main window where we can navigate in the 3D space and interact with the represented objects.

On the right it's possible to see, on the top, a navigation panel whereby we can use a tree structured map of the represented model to navigate through the different layers and select, hide and show the different elements displayed. On the bottom right corner is placed a panel where are shown the properties of the selected objects.

Visualization is supposed to be diversified for three different levels of detail, according to increasing approach in the navigation window:

- **LoD1 visualization:** Only a low detailed and generic model of the whole building is represented;
- **LoD2 visualization:** Are represented the external elements of the building, as roofs, external walls and external ceilings; clicking on them we can benefit from generic information about the aspect and dimensions of the selected entities;
- **LoD3 visualization:** It's intended to be the most detailed visualization where we can interact also with interior elements. To do this it has been supposed a strategic sequence to properly navigate inside the building:
  - Step A: At a specific zoom level over the building, it appears a panel where we can select a building level and, as a thematic plan view of the storey is shown, we can choose a specific room;
  - Step B: Once selected, a camera view is rendered in the navigation window and, once we select a particular object, on the right we can now use a panel with informations about the room in question and, lower, a 3D map of the building where the selected objects are highlighted.

The information represented are more technical, about energetic and mechanical area of interests.



# 3. PROPOSED METHODOLOGY

### 3.1. GENERAL DESCRIPTION

The fundamental step to reach the supposed objectives defined in "2.2.2. Objectives' definition" is to define a methodology or, in other terms, a sequence of operations aimed at gradually define and improve the BIM model, making it fit with our necessities.

To better conceive, understand and communicate this methodology, it has been necessary to draw a diagram composed of different graphical boxes, each representing a particular conceptual or practical operation during the workflow. This diagram is represented in Fig. 3–1.

The diagram is supposed to be read from the top to the bottom, following the logical connections from the START to the END. The rectangular boxes represent the main steps of the methodological path; the curved rectangular boxes indicate specific sub operations; the rhomboidal boxes state for decisional and control points.

It is divided into three main groups, representing three different thematic stages in the operations flow. These groups are:

- **Planning phase:** it includes all the decisional steps aimed at defining the starting point;
- **BIM phase:** it includes all the operations made using the Revit software, till the exportation phase;
- **DIM phase:** it starts from the conversion in the glTF format and ends with the final visualization on the virtual globe.

The orange highlighted boxes represents finally fundamental steps in the standardization of the methodology, described more accurately in the following specific paragraphs.

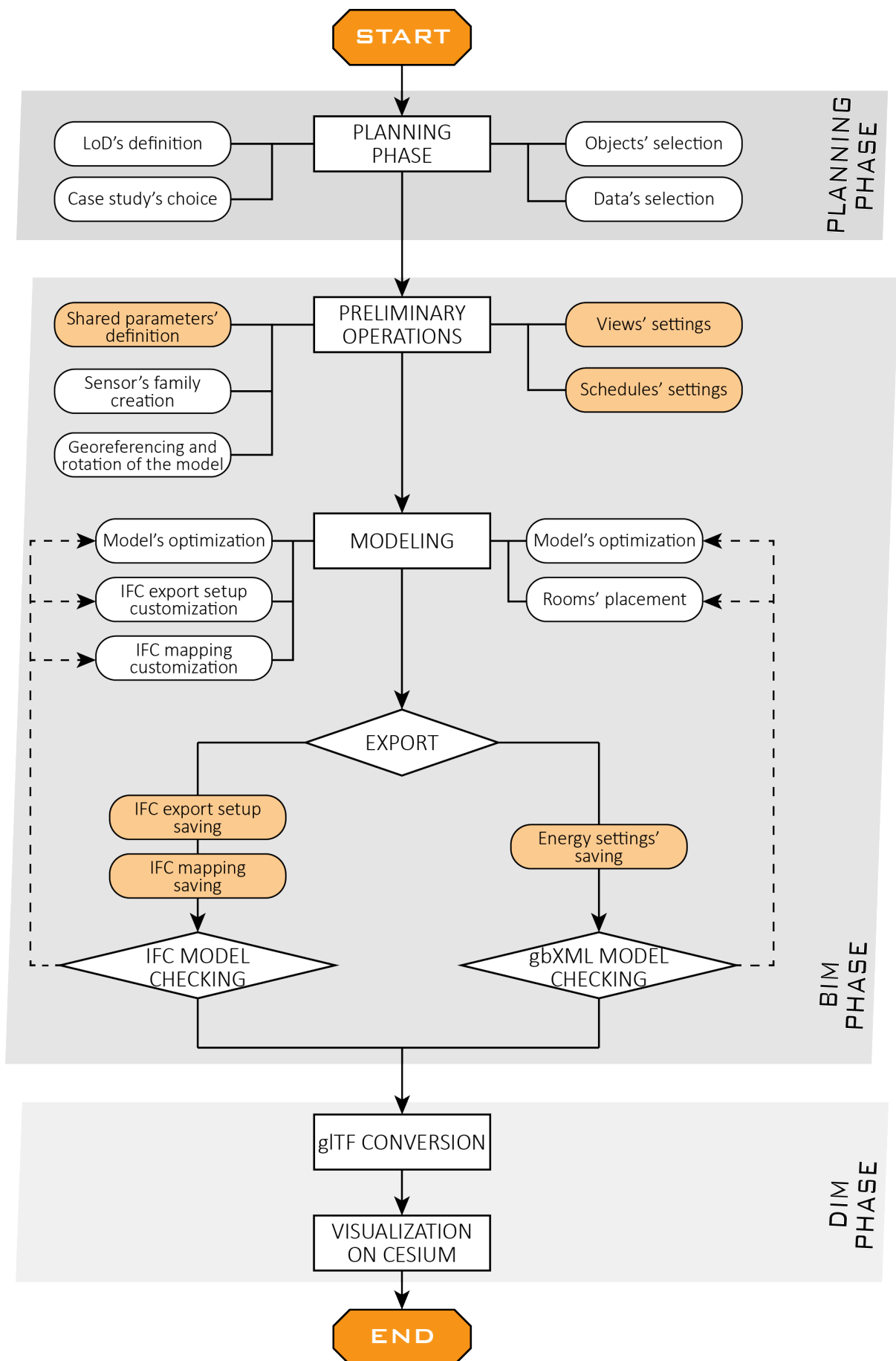


Fig. 3-1 - Diagram of the procedure

## 3.2. PLANNING PHASE

This section consider all the preliminary operations aimed at define the aspect the outcoming model must have.

### Objects' selection

First of all it's necessary to define which of the multitude of the BIM objects we want to be exported. This requires to do a selection among those objects that are supposed to have a relevant role in the final use experience, in terms both of geometrical and analytical importance. According to these premises all the decorative and purely structural elements have been rejected since they are useless or not visible.

This selection also involves a reduction of the 3D model's weight in terms of memory, allowing a better handling in the processes of video rendering.

### Data's selection

Then, it's indispensable to define the informations we want to extrapolate from the BIM model, especially those attached to the objects previously listed above. Here too the work to be done is to recognise the data inherent to the final user necessities, ignoring all the information that are useless for the project's purposes.

### LoD definition

The third problem to face is the definition of the Levels of Detail (LoD), which represent the capability of a 3D model to increase or decrease the complexity of its geometries, as it moves away or it approaches to the viewer or according to a relevance parameter. This feature increases the rendering efficiency, by decreasing the workload on the hardware involved in the visualization and reducing the quality of the objects representation, but without making this visible because of the greater distance.

However, the definition of LoD isn't reserved only to a graphical context but it's nowadays applied also to different field of application. In this particular case, LoD is applied also to show the BIM information according to a specific graphical LoD. In particular it was made a selection among all the data selected in the paragraph above, in order to show for each object and for each LoD representation appropriate informations with, when possible, the opportunity of having a direct reply from the graphical representation.

Tab. 3-1 - LoD definition

	DESCRIPTION	VISUALIZED OBJECTS	VISUALIZED DATA
LOD 1	<ul style="list-style-type: none"><li>• LOW DETAIL</li><li>• ONLY MASSES VISUALIZATION</li></ul>	MASSES	% opaque surface
			% transparent surface
			Building energy certification
			Building typology
			Building use
			Construction period
			Electricity supply
			Location
			Name
			Occupancy number
LOD 2	<ul style="list-style-type: none"><li>• MEDIUM DETAIL</li><li>• GENERIC INFORMATIONS</li><li>• BIM OBJECTS' SELECTION</li><li>• "SHOW/HIDE" OPTION</li></ul>	WALLS	Orientation
			Renewable energy
			Heating supply
			S/V
			CASE_NAME
		CEILINGS	Total annual measured energy consumption
			Total annual simulated energy consumption
			CASE NUMBER
			USE
			TC_sensor
WALLS	HOST_ID		
	Famiglia		
	Tipo		
	Funzione		
	Area		
CEILINGS	Volume		
	Vincolo di base		
	Vincolo parte superiore		
	Altezza non collegata		
	Famiglia		
WALLS	Tipo		
	Livello		
	Perimetro		
	Area		
	Volume		

# LOD 3

- HIGH DETAIL
- DETAILED AND TECHNICAL INFORMATION
- INTERNAL VISUALIZATION AND NAVIGATION

## ROOFS

Famiglia  
Tipo  
Volume  
Area  
Inclinazione  
Livello di base

## WALLS

Famiglia  
Tipo  
Larghezza  
Coefficiente di scambio termico  
REI  
Esposizione

## CEILINGS

Famiglia  
Tipo  
Spessore  
Coefficiente di scambio termico  
REI

## ROOFS

Famiglia  
Tipo  
Spessore  
Coefficiente di scambio termico

## STAIRS

Famiglia  
Tipo  
Livello di base  
Livello superiore

## RAILINGS

(Only visualization)

## DOORS

Famiglia  
Tipo  
Livello  
Da locale: codice locale  
A locale: codice locale  
Altezza  
Larghezza  
Area  
Emergenza  
REI  
Codice



		WINDOWS	Famiglia Tipo livello Da locale: codice locale Altezza Larghezza Area
		SENSORS	Famiglia Tipo Contrassegno Livello Consumo elettrico Potenza elettrica URL istanza
		ROOMS	Nome Numero Codice Locale Utilizzo locale Categoria locale Tipologia locale Direzione Occupanti Area Volume Capacità

Another important step in this stage of the process is the identification of the case study, between those involved in the EeB project and the choice has fallen upon the Scuola Materna Rodari.

Case study's choice

The reason of this selection is due mainly to the fact this is the only one in which real sensors have been placed and this gives the opportunity to improve the virtual modeling and analysis with real data.

Another important reason is because, since the present project starts from pre built BIM models, that of the Scuola Materna Rodari is the most fitting with our needs and objectives. It has in fact a more accurate structure which gives the possibility of getting since the start good results in terms of visualization, exportation and richness of data, thus avoiding time and work spreading for optimizing it and correcting errors.

Tab. 3-2 - Case studies' comparison

	SCUOLA MATERNA RODARI	SCUOLA MEDIA STATALE NICOLI	BIBLIOTECA CIVIVA ARCHIMEDE
GBXML EXPORTATION	✓	✗	✓
IFC EXPORTATION	✓	✓	✗
IN PLACE SENSORS	✓	✗	✗
RICHNESS OF ATTRIBUTES	✓	✓	✗
ACCURATE MODELING	✓	✗	✓

## 3.3. BIM PHASE

### 3.3.1. PRELIMINARY OPERATIONS

#### 3.3.1.1. CREATION OF SENSORS' FAMILY

The first step of this stage of the methodology, is the creation on Revit of the sensors that must be applied on the 3D model and for this purpose it has been created a Revit family called "Sensore cubico" (referring to the geometric representation of this object).

We have initially to click on the home "R" button and then on "New" and "Family". We have now to choose which category of family we want to create and the most suitable option is the "Hosted data device" in order to allow the application of these devices over generic surface, either walls or ceilings.

The sensor is made making a simple extrusion from the hosting surface and suddenly managing its properties going through the "Create" and then "Family Type".

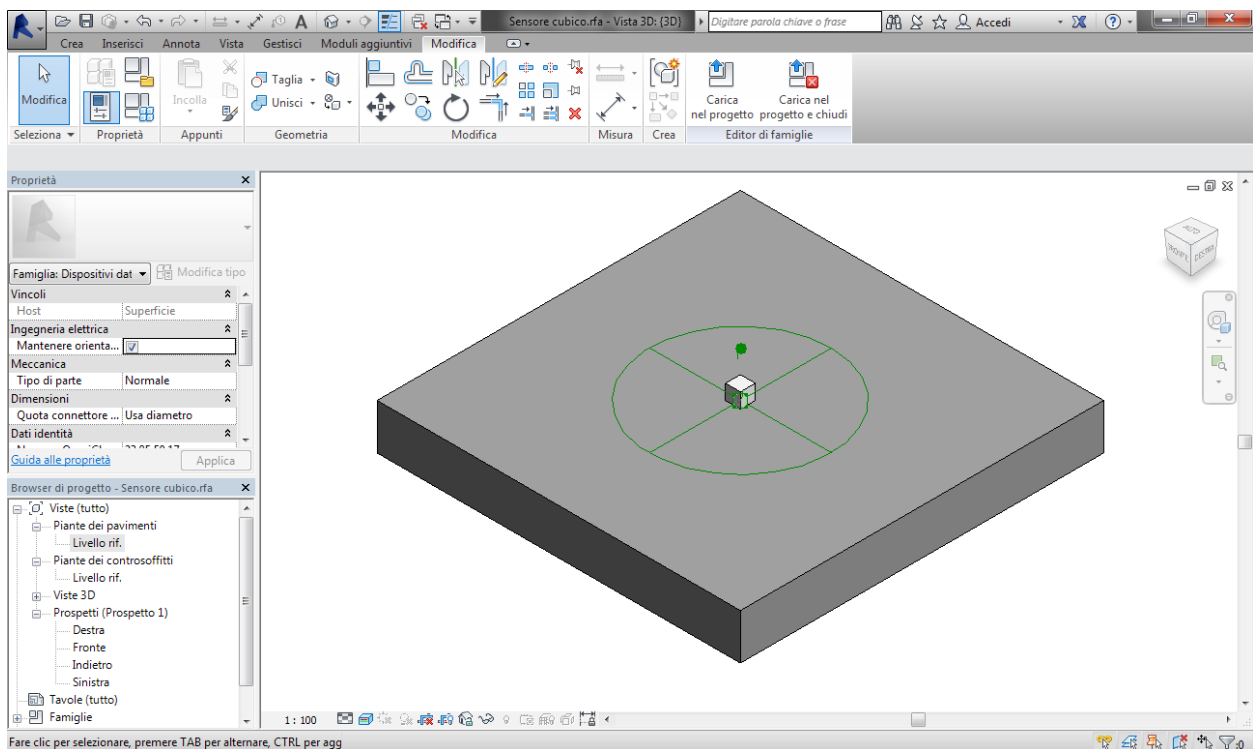


Fig. 3-2 - Sensors' family editing

These sensor have to be placed for the EeB project purpose in specific rooms where the real ones were effectively applied.

### 3.3.1.2. CREATION, SAVING AND OPENING OF A DEDICATED PROJECT TEMPLATE

The first step to accomplish in this preliminary modeling phase is the creation of a customized project template.

This operation corresponds to the definition of a blank default revit model, with well determinate settings, which can be used as starting point for a generic revit project. When we want to start a new project, in fact, the first thing we must do is to choice the template the most corresponding to our needs. Revit has itself a set of industry templates, as "architectural" or "structural", which differ in terms of view settings, unity of measurement, in place families and many other things.

In terms of standards definition, this operation has a leading role since allows to start the modeling phase with a pre defined environment in which the main generic properties are already set and fixed, and so the modeller can work focusing only on the project, without worrying about several generic setting which are already settled.

The common sharing of the same template project is essential when several revit models have to be created with the same characteristics, like in the case of the EeB and the DIMMER projects. In this way all the projects will look in the same way, using the same general settings and nomenclature, avoiding issues due to a different, incoherent or conflicting preparation of them.

Creating a good template is essential to working smarter and faster, becoming more productive and saving valuable time.

In general, a project template can be defined by different aspects, such as naming standards, annotations, materials, object stiles, lineweights, etc.

In this specific case, a template project has been created adding first of all the sensor's family created as described in the previous paragraph. This allows the operator to have direct access to this kind object, without the need of make him require or search it from external sources.

Other customisations include the creation of a specific 3D view and schedules, as defined more accurately in "3.3.3.2.1. Preliminary operations on Revit".

These operations are dedicated to the attempt of creating a standard exportation of the Revit to the IFC format. Note that the schedules' setting can be defined a priori, without successive customization needs, but the 3D view, even if view's are defined, we need to improve the visualization by hiding and show specific objects, according to the specific project needs and following the guidelines proposed in the paragraph.

Once we've defined the project template, we can save it as a .rte file and successively open it in order to start the modeling of a new project.



Fig. 3–3 - Project template file

### 3.3.1.3. GEOREFERENCING AND ROTATION OF THE MODEL

The situation at the beginning of this procedure is the one represented in the image below.

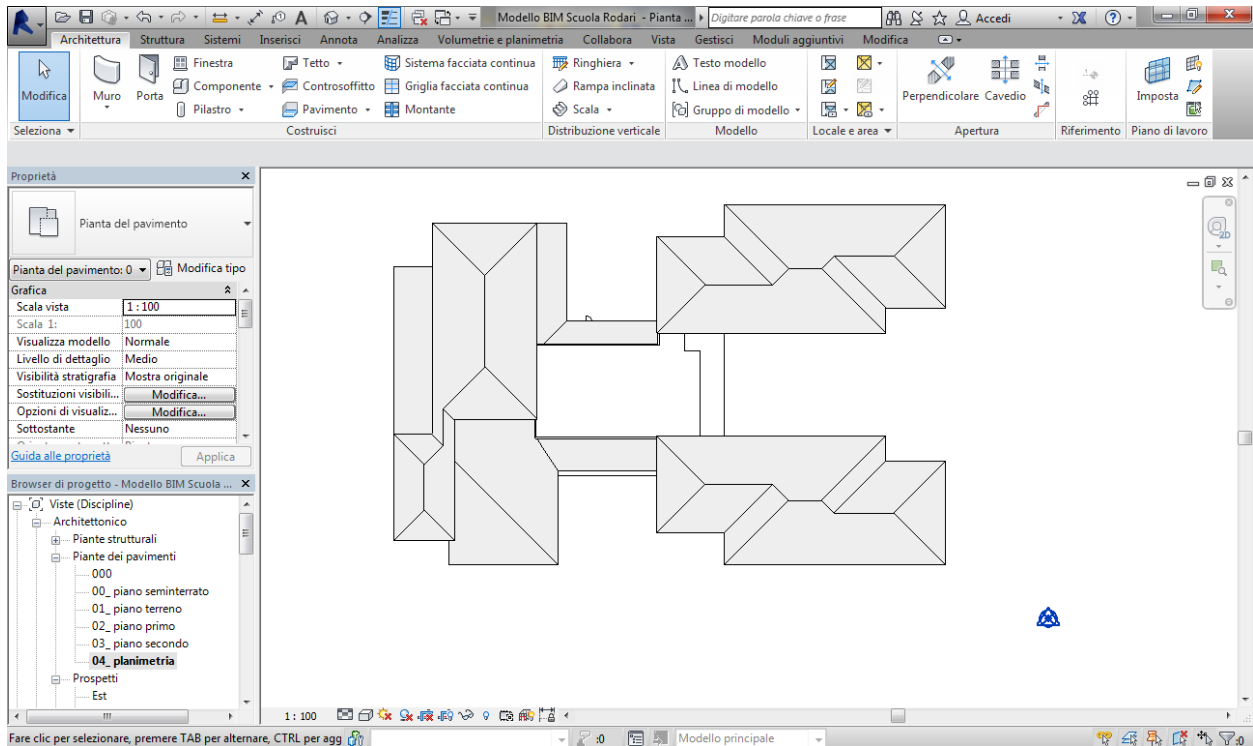




Fig. 3-4 - Georeferencing (step 1)

In the lower right corner of the window it's possible to see that the Survey Point and the Project Base Point aren't combined with the 3D model.

Georeferencing

The Project Base Point  defines the origin (0,0,0) of the coordinate system of the project itself and can be used for positioning the building on the planimetry or for identifying design elements of a building during construction. Clicking on it the coordinates and elevation spot are displayed, referring to the coordinate system of the project. The Survey Point  represents a known real physical point, such as a geodetic detection mark, and it is used to correctly orient the geometry of the building to a different coordinate system, such as that used in a particular civil engineering application. Clicking on it it's possible to show the geographic coordinates attached to this point.

As it appears in Fig. 3-4, the 3D model is placed randomly in the physical space and so we have to give it a well defined positioning.

The thing to do is first of all to obtain the geographic coordinate of a specific point of the model and to succeed in this we used the Google Earth Pro application.

After we choose a reference point along the perimeter of the building, we must place, more accurately as possible, a "placeholder" represented by a yellow pin. Once did this, it will appear a window which give us the coordinate of the

selected point.

We can highlight the fact that the fact that the coordinate system used by Google Earth Pro, World Geodetic System 1984 (WGS84), is the same in use in Revit; so this will prevent from evident placement errors.

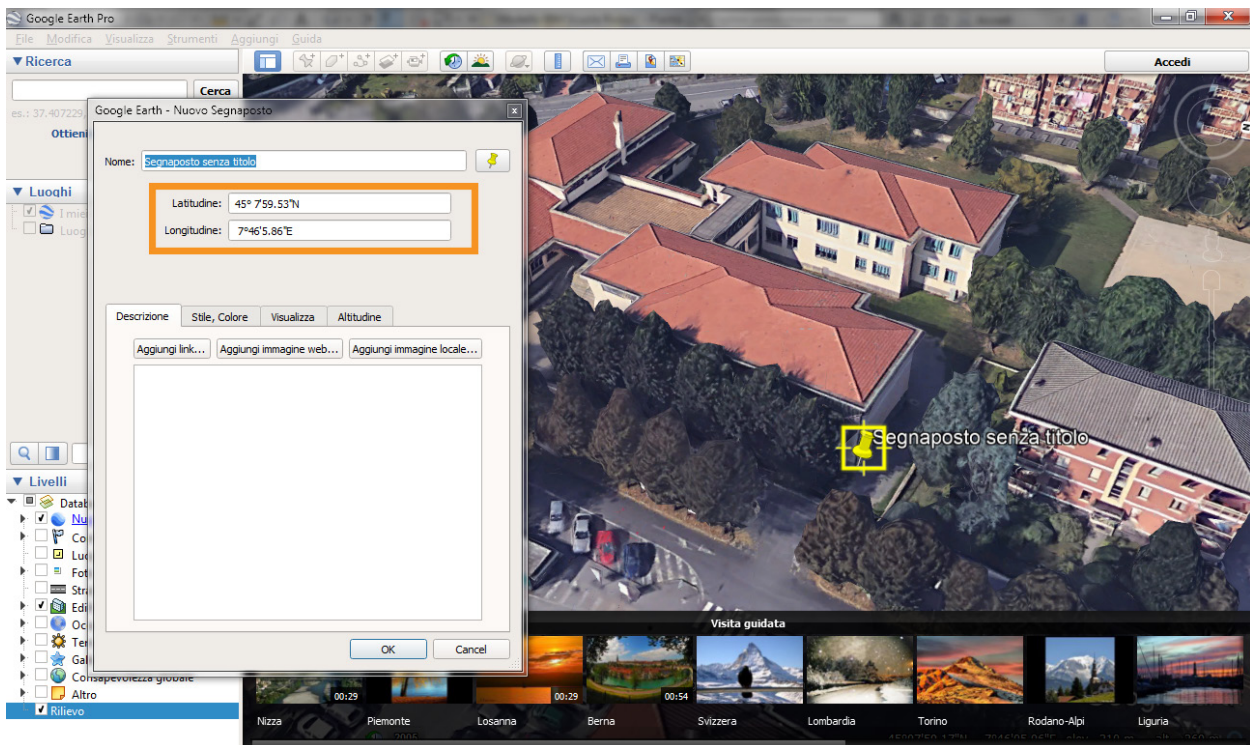


Fig. 3-5 - Georeferencing (step 2)

Once we obtained these values, we must open the "Manage" tab in Revit, click on the "Position" icon and then it will appear a window with an empty bar (see Fig. 3-6) where we have to paste our coordinates in the form "(Latitude), (Longitude)".

In the map below the bar, the red pin, representing the position of the survey point, will move towards the geographic reference point managed in Google Earth.

We have to specify that it's possible to do this operation of placing the pin, directly from this Revit window but because of the bad quality of the map visualization and of the low precision which derives from it, it's better to lean on a more powerful and specific tool like Google Earth.



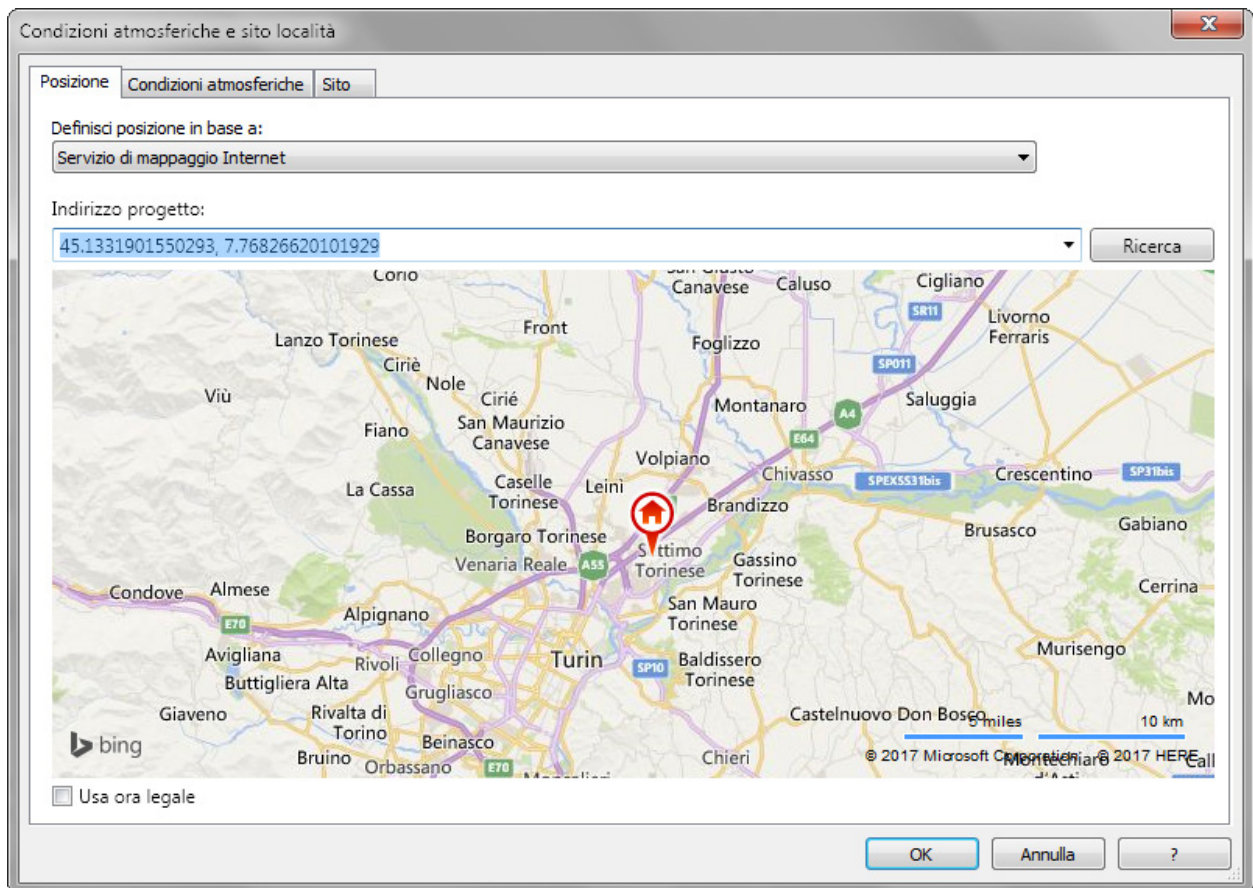





Fig. 3-6 - Georeferencing (step 3)

Clicking on the "OK" button the window will close and the Survey Point is finally associated to well defined geographic coordinates. But now we have to associate them to a specific point of the 3D model, the same one chosen as reference in Google Earth. To do this is necessary to click on the Project Base Point and a "clip" icon will appear in this form ; clicking on it, it will change in , and that means we have unlocked its positioning, which is for default constrained to the Survey Point.

Now we can move the Project Base Point to the reference point and we can see that it moves away from the Survey Point. Once placed, we have to lock it again clicking on , in order to attach him to the building. At this point, however, the virtual model is still placed randomly and the Survey Point, which includes the geographic coordinate, isn't attached to the model.

To prevent this, we must click again on the Project Base Point. On its right they will appear informations about its N/S and E/W placement in relative coordinates, and as we can see they are initially not well defined. Clicking on these values we can manage each one to a zero value and the Project Base Point will immediately move and overlap the Survey Point. The reference point of the 3D model is now attached to the Survey Point and so the geographic coordinates and this means that the 3D model is attached to the correct geographic coordinates.

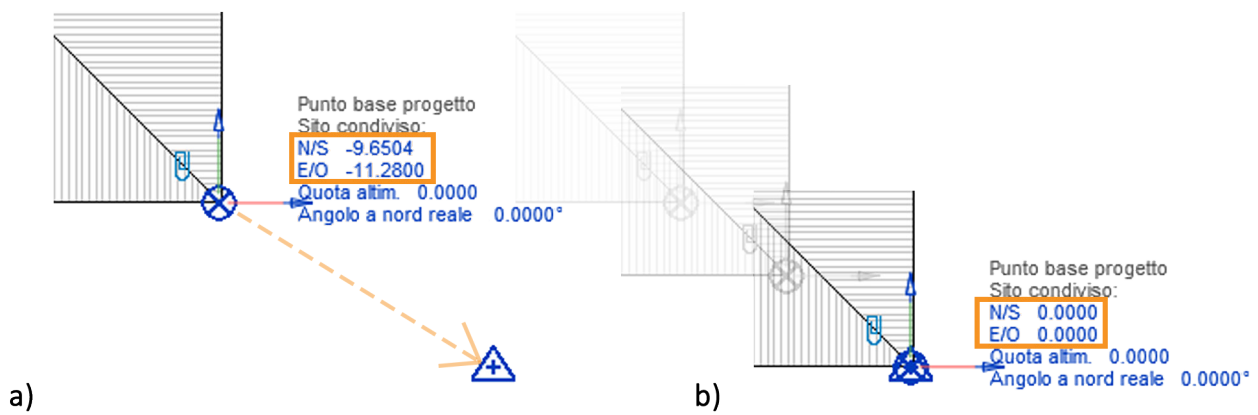


Fig. 3-7 - Georeferencing (step 4)

#### Real north rotation

Once we correctly placed the 3D model in the geographic coordinate system, we can proceed by rotating the project north in order to coincide it with the real one.

To get information about the rotation angle we can use a satellite view or, in order to be more accurate, we can get technical maps. In this case it has been downloaded from the website of the Municipality of Settimo Torinese a DWF file that, once opened with Autocad, allows to calculate and then save the angle, taking a reference point.

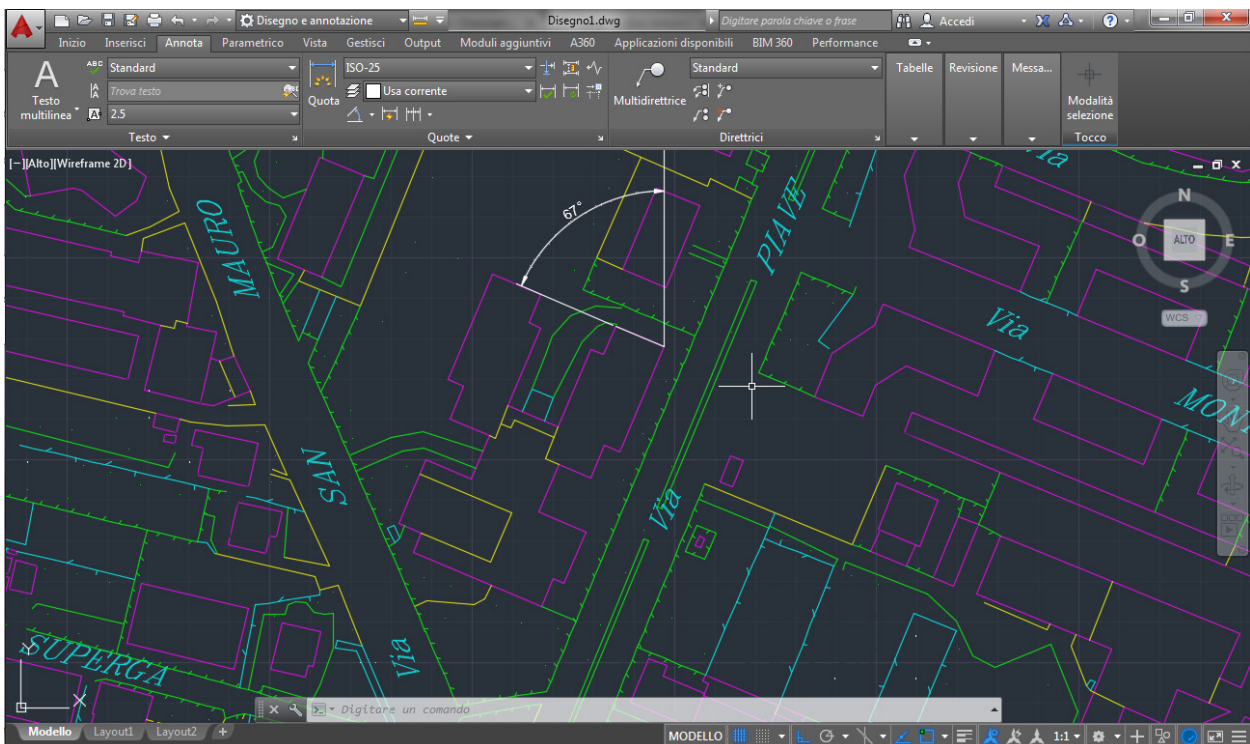


Fig. 3-8 - Real north rotation (step 1)

Returning on the Revit interface and opening a generic plan view, we have to manage the property window setting the "Orientation" option to "Real north". We can now go to the "Manage" tab and click on "Position" and then "Rotate true north". We can now place the rotation center at the reference point took to



calculate the angle on autocad, and so setting the rotation angle manually.

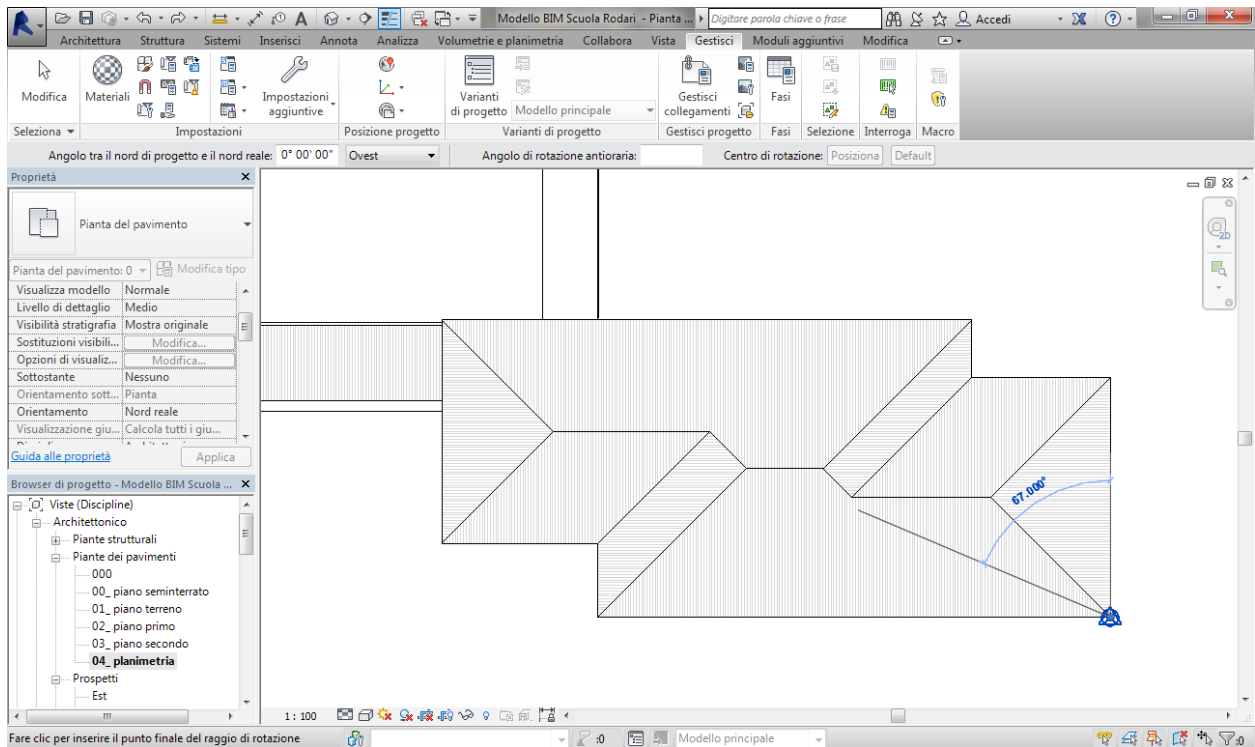


Fig. 3-9 - Real north rotation (step 2)

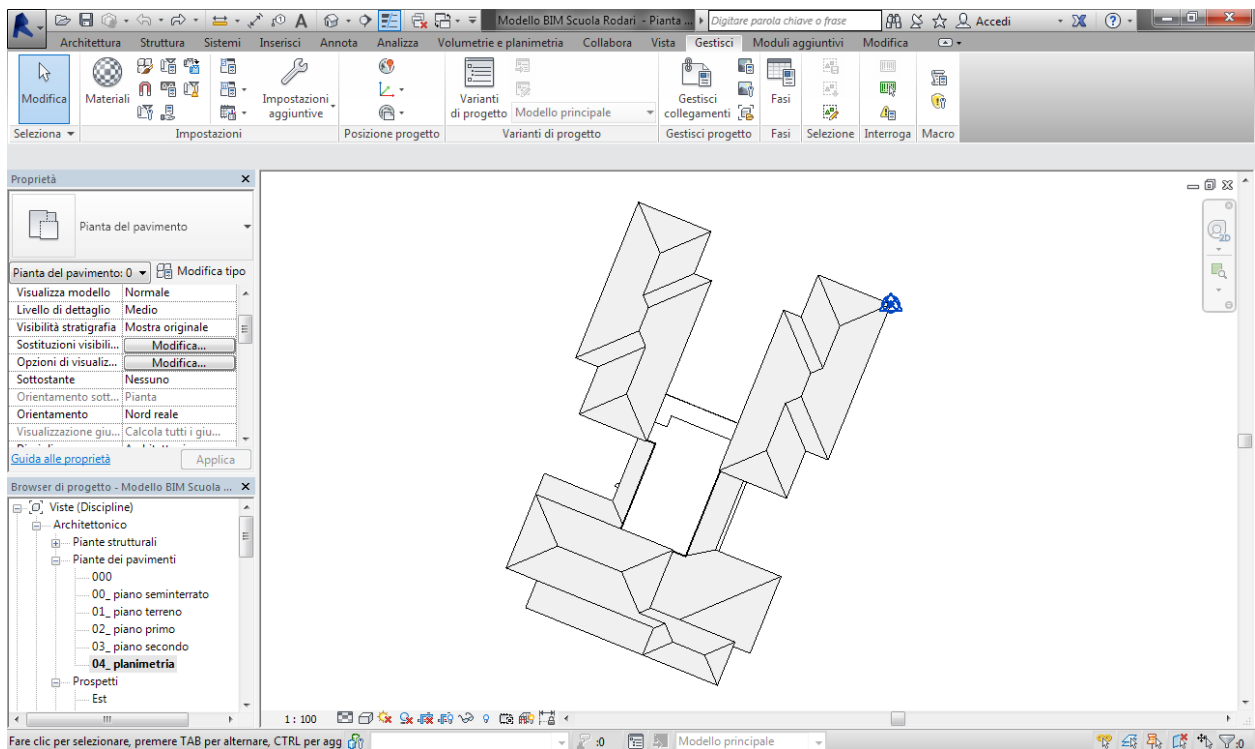


Fig. 3-10 - Real north rotation (step 3)

#### 3.3.1.4. CREATION OF A SHARED PARAMETER FILE

Creation of a Shared parameters file

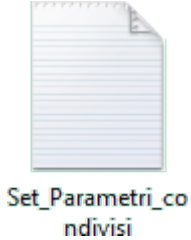


Fig. 3–11 - Shared parameters file

In order to facilitate the standardisation of the operations in Revit, it's very useful to benefit from the creation of a shared parameter file.

This step is fundamental, not so much for the exportation (especially the RVT-IFC conversion), but in the optic of the successive operation of parsing the exchange models. It's very important in fact that the attributes setting in Revit follows a well defined encoding which can be instructed by the shared parameters functionality.

This file must include the list of all those attributes that we have defined above for each object, in order to have a unique nomenclature and to avoid the presence of redundant attributes.

This file can be suddenly loaded inside the Revit project template defined in "3.3.1.2. Creation, saving and opening of a dedicated project template", so that it can be accessible directly from the start of a new project based on that template.

## 3.3.2. RVT - GBXML EXPORTATION

### 3.3.2.1. GBXML FORMAT

#### 3.3.2.1.1. INTRODUCTION

##### MAIN FEATURES

gbXML, which stands for "Green Building XML schema", is an open schema based on XML language and developed to facilitate the transfer of building information, stored in building information models, enabling integrated interoperability between building design models and a wide variety of engineering analysis tools and models available today.

Today, gbXML has the industry support and wide adoption by leading Building Information Modeling (BIM) vendors including Autodesk, Trimble, Graphisoft, and Bentley. With the development of export and import capabilities in over 40 engineering and analysis modeling tools, gbXML has become a industry standard schema. Its use dramatically simplifies the transfer of building information to and from architectural and engineering models, eliminating the need for time consuming plan take-offs. This removes a significant cost barrier to designing sustainable and energy efficient buildings. It enables building design teams to truly collaborate and realize the potential benefits of Building Information Modeling.

Therefore, this standard is designed for including only the information needed to support energy analysis being commonly used as input for energy simulation software.

##### HISTORY

In 1999, Green Building Studio, with the collaboration of the California Energy Commission PIER Program and Pacific Gas and Electric, started the development of the gbXML project.

In June of 2000, the first gbXML schema was submitted and shortly thereafter, gbXML became the draft schema for the Building Performance & Analysis Working Group.

In 2008 Green Building Studio was acquired by Autodesk.

In 2009, gbXML was spun off from Green Building Studio to become a stand-alone entity. Since this moment, gbXML project experienced a faster development: funding was secured, the schema was drastically improved, a new website was launched, and a community of thousands of architects, engineers, and energy



Fig. 3-12 - gbXML logo

modelers attended live webinars explaining the benefits of gbXML.

Today, gbXML is funded by organizations such as the U.S. Department of Energy, the National Renewable Energy Lab (NREL), Autodesk, Bentley Systems, and others.

The employment of this exchange format allows building designers to focus on creating environmentally responsible buildings that use intelligent technologies to meet their client's needs at the lowest cost possible. Helping realize the promise of Building Information Modeling, gbXML allows intelligent solutions for the design, certification, operation, maintenance, and recycling of buildings.

### 3.3.2.1.2. SYNTAX AND STRUCTURE

#### XML INHERITANCE

##### XML as markup language

XML, abbreviation for "extensible markup language", is a type of computer language that allows software programs to communicate information with little to bigger interaction.

XML was developed by the World Wide Web Consortium (W3C) in 1998, with the purpose of creating a extensible markup language of general use. It derives from the Standard Generalized Markup Language (SGML), which is a standard for defining generalized markup languages for documents, developed by the International Standard Organization (ISO) in 1986. XML, however, differs from it for his specific World Wide Web oriented application.

A markup language is a computer language that uses tags to define elements within a document. It is human-readable, meaning markup files contain standard words, rather than typical programming syntax. "Extensible" means that it's possible to create a set of personalized tags.

This language is normally used for the exchange of documents but also for store and exchange data structures. It is supported by Unicode encoding in order to admit a great interoperability between different human languages, supporting in this way the use of several alphabets and symbols used around the world and in different application fields.

XML is a free domain standard, and this allows the development of several libraries for XML's data manipulation in different programming languages and the capability of a great number of software to read XML models.

Its great interoperability is also granted by the fact it is software and hardware independent, so that it allows the representation of every kind of document independently from his applicative finalities.

##### XML as metalanguage

XML can be also be described as a metalanguage. In fact, such as the grammatical definitions and rules describes a language, XML defines a set of syntactic rules

which formally defines a markup language known as "XML application", like the gbXML schema. Every XML application derives a set of common syntactic characteristics and defines a formal particular syntax.

An XML document is considered to be "well formed" (that is, able to be read and understood by an XML parser) if its format complies with the XML specification, if it is properly marked up, and if elements are properly nested.

## SYNTAX

Looking to the definition of the XML language it's possible to give a proper definition of the gbXML schema, analysing its main entities and the way the building model decomposes into the schema.

Basically, the gbXML exchange format is a set of text string elements, organized in a hierarchical structure and composed of "markup" (or "tag") and "content" entities.

Elements, markups and contents

gbXML documents must contain above all an element called prolog. In this element is possible to read the XML version and the character encoding used in the gbXML model. This element doesn't belong to the gbXML model.

After the declaration of the prolog, the gbXML schema requires the presence of a root element that is the parent of all other elements, in their turn all disposed as child element of the root element in a tree structure. The root element contains the actual gbXML model.

```
<?xml version="1.0" encoding="UTF-16"?>
```

Root Element schema

Markups can be either start-tags or end-tags: the first ones are strings of characters contained in and including the "<[...]>" Structure; end-tags are instead described by the "</[...]>" form. Strings of characters that are not markup are content. Markups describes the structure and the aspect of the document.

An element is a logical document component that either begins with a start-tag and ends with a matching end-tag. The characters between the start-tag and end-tag, if any, are the element's content, and may contain markup, including other elements, which are called child elements. This ability allows XML to support hierarchical structures. Element names describe the content of the element, and the structure describes the relationship between the elements.

XML elements can have attributes, which are particular entities contained in the tag and designed to declare data related to a specific element. Attribute are composed by a declaration and, included in doubles quotes, a content part; the two parts are separated by the "=" symbol. In the gbXML schema these entities are very useful to declare metadata, like ID references between the XML

Attributes

elements.

For a better comprehension of the gbXML syntax we can analyse a simple example referring to a bookstore's list of item.

```
<bookstore>
  <book category="children">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

<title>, <author>, <year>, and <price> have text content because they contain text strings.

<bookstore> and <book> have element contents, because they contain elements. Those elements are called "child elements".

<bookstore> is the Root Element of this XML file.

<book> has an attribute (category="children").

### STRUCTURE

If we open a gbXML model as a text file we will find a tree structure described as below.

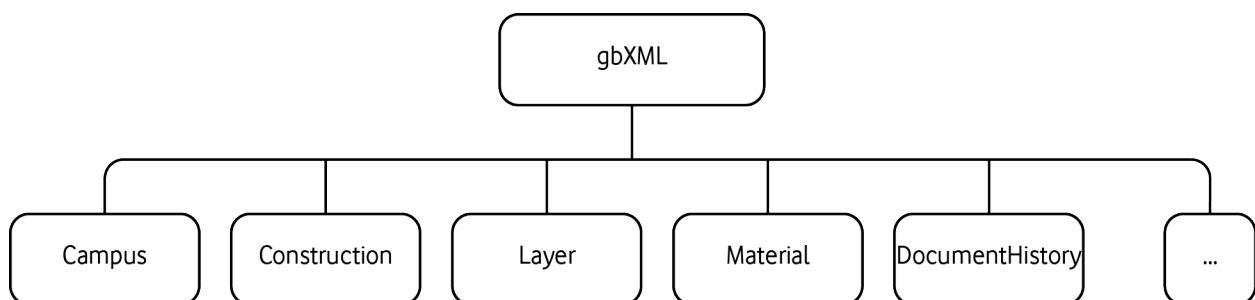


Fig. 3-13 - General structure of the gbXML model

This tree structure is not intended to be a definitive list of all the elements in a gbXML document, just to give a sense of the level of nesting of informations. The elements can in fact be ignored or included in the gbXML model, depending on the export operation. In the case of the exportation made through the default gbXML exporter included in Revit, the only child elements exported will be at most:

- Campus;

- Construction;
- Layer;
- Material;
- DocumentHistory.

The gbXML element, which is the root element of the model, describes with its attributes the units of measurement used in the gbXML model and the gbXML schema version.

```
<gbXML useSIUnitsForResults="true" temperatureUnit="C" lengthUnit="Meters"
  areaUnit="SquareMeters" volumeUnit="CubicMeters" version="0.37" xmlns="http://
  www.gbxml.org/schema">
```

#### gbXML Element Schema

All the geometric information is defined in the Campus element, composed of the global child elements: Location, Building and Surface.

Campus element

The Location element is used to indicate the global position of the building and its orientation from the North.

```
<Location>
  <StationId IDType="WMO">160478_2006</StationId>
  <ZipcodeOrPostalCode>00000</ZipcodeOrPostalCode>
  <Longitude>7.76864</Longitude>
  <Latitude>45.1382</Latitude>
  <Elevation>709.8792</Elevation>
  <CADModelAzimuth>0</CADModelAzimuth>
  <Name>Settimo Torinese, Piem., Italy</Name>
</Location>
```

#### Location Element Schema

The Building element describes the spaces or volumes enclosed by surfaces. It decomposes in the child elements Space and BuildingStorey: the first one defines the position of each space in the model, its planar geometry, its shell geometry and its relation with the surfaces which compose it; the second one defines the height and the planar geometry of the building storeys present in the model. The CADObjectId element contained into the Space element present as its content a set of six numbers: this is the CAD object ID identifier of the Revit instance and it's very important in the development of the project because it represents a univocal bridge between the Revit model, the gbXML model and the IFC model.

Building element

The Surface elements describe every surface generated in the gbXML exportation, defining their borders, their orientation; they may eventually contain the presence of the child element Opening to describe the same characteristics of a opening element attached to those surfaces. The surfaces, as declared by the SurfaceType attribute, can be either of these categories:

Surface element

- ExteriorWall;
- InteriorWall;

- UndergroundWall;
- InteriorFloor;
- SlabOnGrade;
- RaisedFloor;
- UndergroundSlab;
- Roof;
- Opening;
- Air;
- Shade.

The type of surface is figured out depending on the source element and the number of space adjacencies. If there is no associate source element and no space adjacencies, it will have a type of Shade. If there are any space adjacencies, it will have a type of Air.

In terms of geometry, gbXML only represents rectangular shapes, simplifying the geometry so that it can be used as input for most energy analysis tools. The characteristic points that define one surface are included in the child CartesianPoint elements that give the model its three-dimensional representation through its three coordinates (x,y,z). These coordinates are relative to the base point of the building model.

The Name child element is automatically generated during the exportation, following a specific encoding which reports synthetically information about the orientation and the spaces to which the surface is related.

The CADObjectId element reports instead the name of the Revit instance associated to the analytical surface in question, specifying the family name and the type name. It's very important to notice in the content of this element the presence, as pointed out when talking about the Space elements, of a string of six numbers between square parentheses which does the same important job.

```
<Surface surfaceType="ExteriorWall" constructionIdRef="aim7316"
  exposedToSun="true" id="aim7857">
  <AdjacentSpaceId spaceIdRef="aim0441" />
  <RectangularGeometry id="aim7858">
    <Azimuth>180</Azimuth>
    <CartesianPoint>
      <Coordinate>-33.14352</Coordinate>
      <Coordinate>-22.76354</Coordinate>
      <Coordinate>-2.55</Coordinate>
    </CartesianPoint>
    <Tilt>90</Tilt>
    <Width>7.09999990463257</Width>
    <Height>2.54999995231628</Height>
  </RectangularGeometry>
  <PlanarGeometry>
    <PolyLoop>
```



```

<CartesianPoint>
  <Coordinate>-33.14352</Coordinate>
  <Coordinate>-22.76354</Coordinate>
  <Coordinate>-2.55</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>-26.04352</Coordinate>
  <Coordinate>-22.76354</Coordinate>
  <Coordinate>-2.55</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>-26.04352</Coordinate>
  <Coordinate>-22.76354</Coordinate>
  <Coordinate>0</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>-33.14352</Coordinate>
  <Coordinate>-22.76354</Coordinate>
  <Coordinate>0</Coordinate>
</CartesianPoint>
</PolyLoop>
</PlanarGeometry>
<CADObjectId>Muro di base: EXT40 [1693541]</CADObjectId>
<Name>S-1-E-W-2</Name>
</Surface>

```

Example of Surface element schema

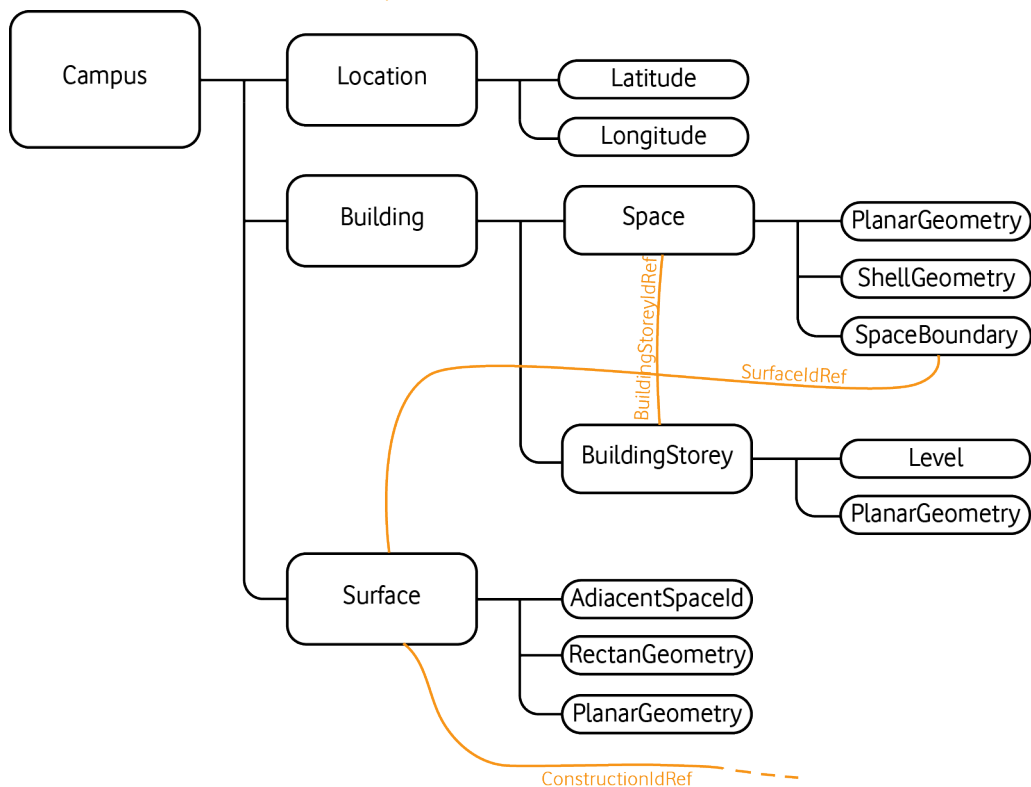


Fig. 3-14 - Campus element definition

Every Surface element, through its constructionIdRef attribute, is related to a specific Construction element. This element is a combination of layers, such as a wall or a roof. It contains several attributes useful for the definition of the thermal and mechanical characteristics of the related Surface element. If the "Export Defaults" under Energy Settings in Project Information is not checked,

Construction element,  
Layer element and Ma-  
terial element

this element is not exported and so its child elements Layer and Material. Each Construction element is related through the layerIdRef attribute to a Layer element, which consists of a orderly set of MaterialId child elements. These refer in their turn to a specific Material element where it's possible to collect further information about the thermal and mechanical properties of the materials.

```
<Construction id="aim7262">
  <U-value unit="WPerSquareMeterK">1.346738</U-value>
  <Absorptance unit="Fraction" type="ExtIR">0.7</Absorptance>
  <Roughness value="Rough" />
  <LayerId layerIdRef="aim7266" />
  <Name>Pavimento: INT35</Name>
</Construction>
<Layer id="aim7266">
  <MaterialId materialIdRef="aim7267" />
  <MaterialId materialIdRef="aim7274" />
  <MaterialId materialIdRef="aim7281" />
  <MaterialId materialIdRef="aim7288" />
  <MaterialId materialIdRef="aim7295" />
</Layer>
<Material id="aim7274">
  <Name>Sabbia e Calcestruzzo: 0.07 [m]</Name>
  <R-value unit="SquareMeterKPerW">0.3349282</R-value>
  <Thickness unit="Meters">0.07</Thickness>
  <Conductivity unit="WPerMeterK">0.209</Conductivity>
  <Density unit="KgPerCubicM">950</Density>
  <SpecificHeat unit="JPerKgK">657</SpecificHeat>
</Material>
<Material id="aim7281">
  <Name>Calcestruzzo, gettato in opera: 0.06 [m]</Name>
  <R-value unit="SquareMeterKPerW">0.05172414</R-value>
  <Thickness unit="Meters">0.06</Thickness>
  <Conductivity unit="WPerMeterK">1.16</Conductivity>
  <Density unit="KgPerCubicM">2000</Density>
  <SpecificHeat unit="JPerKgK">870</SpecificHeat>
</Material>
<Material id="aim7288">
  <Name>Laterizio pav: 0.18 [m]</Name>
  <R-value unit="SquareMeterKPerW">0.3</R-value>
  <Thickness unit="Meters">0.18</Thickness>
  <Conductivity unit="WPerMeterK">0.6</Conductivity>
  <Density unit="KgPerCubicM">1800</Density>
  <SpecificHeat unit="JPerKgK">870</SpecificHeat>
</Material>
<Material id="aim7295">
  <Name>Intonaco interno: 0.02 [m]</Name>
  <R-value unit="SquareMeterKPerW">0.03921569</R-value>
  <Thickness unit="Meters">0.02</Thickness>
  <Conductivity unit="WPerMeterK">0.51</Conductivity>
  <Density unit="KgPerCubicM">1120</Density>
  <SpecificHeat unit="JPerKgK">960</SpecificHeat>
</Material>
```

Example of Construction Element, Layer Element and Material Element schema

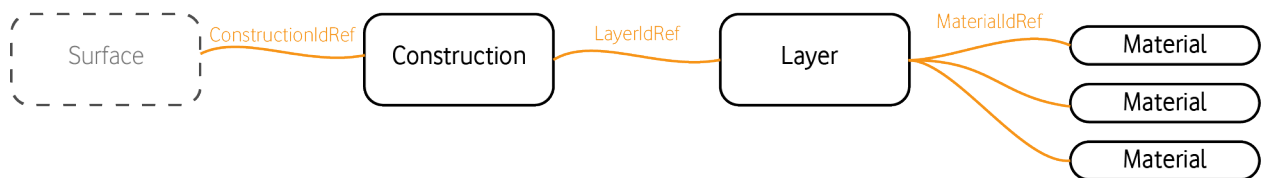


Fig. 3–15 - Construction element, Layer element and Material element definition

The DocumentHistory element gather informations about the people and programs that created and modified the gbXML file. It's important to notice the CADModelId child element which gives us a univocal identifier of the Revit model, used to map a CAD model or BIM to its corresponding gbXML file, represented by a hexadecimal string of characters.

DocumentHistory element

```

<DocumentHistory>
  <ProgramInfo id="aim0004">
    <CompanyName>Autodesk, Inc.</CompanyName>
    <ProductName>Autodesk Revit 2016 BEES</ProductName>
    <Version>2016 20150714_1515(x64)</Version>
    <Platform>Microsoft Windows 7</Platform>
    <ProjectEntity>
      <URI>file:///C:/Dropbox/TESI/Casi studio/Modello BIM Scuola Rodari .rvt</URI>
      <GUID entity="File">306389db-f5a9-4e5e-b9ac-035c2cca6b2e</GUID>
    </ProjectEntity>
  </ProgramInfo>
  <CreatedBy personId="aim0005" programId="aim0004" date="2017-02-14T09:16:19">
    <CADModelId>306389db-f5a9-4e5e-b9ac-035c2cca6b2e</CADModelId>
  </CreatedBy>
  <PersonInfo id="aim0005" />
</DocumentHistory>
  
```

DocumentHistory schema

### 3.3.2.2. CUSTOMIZATION OF THE EXPORTATION: CREATING A STANDARD

#### 3.3.2.2.1. PRELIMINARY OPERATIONS ON REVIT

##### MODEL'S GEOMETRIES OPTIMIZATION

This section explains the most commonly encountered issues with Revit models which lead to incorrect geometry within the gbXML file, and gives solutions in each case. The best way to ensure a good gbXML import from Revit is to have these guidelines in mind from the beginning of the project. Correcting an existing Revit model is much more complicated and time-consuming.

The aim of this section is to ensure that when rooms are applied to the Revit model the resulting gbXML will have a complete set of surfaces and all the

surfaces will have the correct type.

Ignoring these rules may cause confused and incorrect geometry in the gbXML such as missing walls, walls which have no connection to the building, adjacent rooms failing to join together, and internal surfaces exposed to the outside air.

## Rooms

The first operation to do is to verify that all rooms are properly contained and that all rooms link correctly to neighbouring rooms or to external elements of the building. Rooms are the fundamental entities in the gbXML export operations and the preliminary correct disposition of these will allow to proceed with the first export attempt and the gradual optimization of the Revit model according to the issues highlighted along the procedure. Any part of the building which is not occupied by a room will not be exported correctly to the gbXML model. If a room is obstructed in any way from touching the surfaces associated with it, the data exported to the gbXML will be incorrect.

The first expedient is to fill every volume in the building with a room entity, even if the concerned space isn't intended to have particular relevance or dimensions, even in case of wall voids. This operation is necessary to export a completely defined geometric shell of the building, without void elements, especially in the exterior facade. This can be done by creating a new specific room or by expanding a neighbouring pre existent room.

Another important operation is to verify the vertical borders of the rooms: it's fundamental that the lower and the upper limits of the room entities correspond respectively to (at least) the finishing of the floor and the ceiling, taking care there isn't a overlapping with other room entities in the same or other building levels. This procedure allows, during the exportation, the creation of a complete set of coherent analytical surfaces.

Finally, it's important to check the correct disposition of room separation lines. These should only be used to separate one room from another but in all other instances room separation lines should be deleted. Room separation lines should never overlap walls or other room separation lines. If they do then this will show up as a warning within Revit, so check that any such warnings have been resolved before exporting the model.

## Roof and floor footprints

When drawing the building's roof and floors, we must bear in mind that any gaps in the roof will allow Revit rooms to leak from the building and, in case of gbXML exporting, the room below the roof will generate undesired Air surfaces. In order to ensure that the roof footprints are sufficient to fully cover the rooms below them it can be useful to increase the upper limits of the rooms on the top floor and then go to the gbXML export dialog window. In this way it can be possible to check the presence of roof's and ceiling's gap, highlighted by the presence of vertical flat surfaces leaking out the room's upper border.

In case of parallel and overlapping building elements (like walls, roofs and ceilings) it's recommended to replace these with a unique equivalent entity. This precaution will prevent the undesired separation of two rooms that are in fact linked each other, since there is not a connection between the two building elements on Revit.

Parallel and overlapping elements

All columns must be made non-room bounding. Room bounding columns create external links within the building, meaning the column surfaces are treated as if they were exposed to the outside air. Room-bounding columns can also lead to confused geometry being output to the gbXML, such as missing areas. Where columns form part of a wall between two rooms and removing the column would cause the rooms to occupy the same area, the wall should be extended to fill the gap left by the column.

Columns

In the end, some geometric issues can be prevented by the use of two Revit commands: "join geometries" and "wall joins".

"Join geometries" and "wall joins" commands

The first one must be used to improve the connection between different building elements, allowing the analytical surfaces to better communicate each others. This command permit also to notice if two elements aren't well connected such as a ceiling with a wall.

The second one, instead, allows to manage the intersection between walls, optimizing the creation of the correct analytical surfaces also in corner zones, which are frequently a critical zone.

### CREATION OF SENSORS

This paragraph describes the procedure necessary for the geometric exportation of the sensor applied in the case buildings.

Since the gbXML only exports room elements and the analytical surfaces related to these, it was indispensable to find an alternative way to allow the sensors to be seen once the model was exported in gbXML format.

The solution was found by creating fake room entities, following a well defined procedure described below.

The first thing to do is to create the "container" of the sensor room. Looking at Fig. 3–16, the steps to follow are:

- a) Create a spherical mass element;
- b) Apply wall on top surface;
- c) Apply wall on low surface;
- d) Join the two walls with the "Join geometries" command.

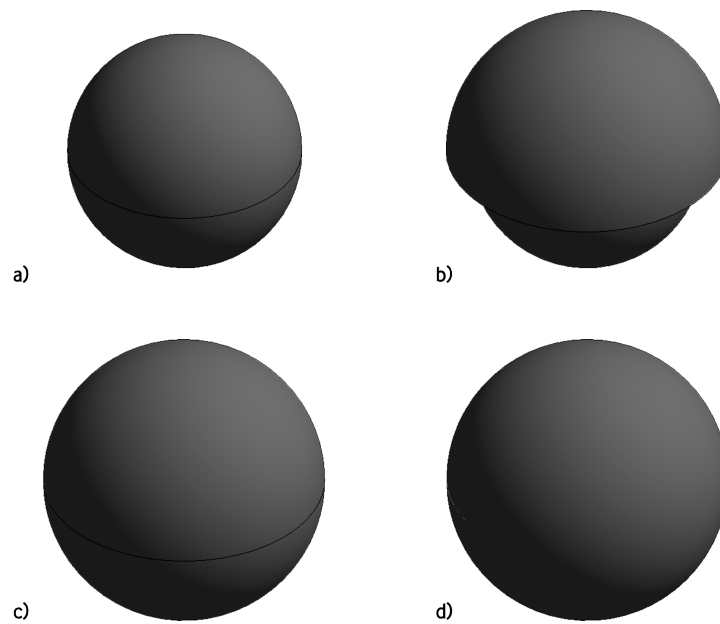


Fig. 3-16 - Creation of the Sensor room's container

Once made this and deleted the mass, we have created the void element to fill with the sensor room. To do this we have to simply use the "Room" command in Revit and click inside the shell we've created; then, going to a section view which intersect the container, we can manage the upper and lower borders of the cylindrical room generated.

The choice of using a sphere as basis for this operation is due to the fact that other geometric support had done some problems, interfering with the other effective room (like creating void element above the sensor). This procedure allows also to use the simpler closed shell possible on Revit, using only two curved walls, avoiding the use of additional ceilings or floors and then the risk of succumb to modeling's error.

The procedure concludes by assigning as name of the sensor room the encoded name of the effective sensor used in the case buildings. This will permit to distinguish the sensors from the other spaces of the building, making possible to interrogate them in the final applications.

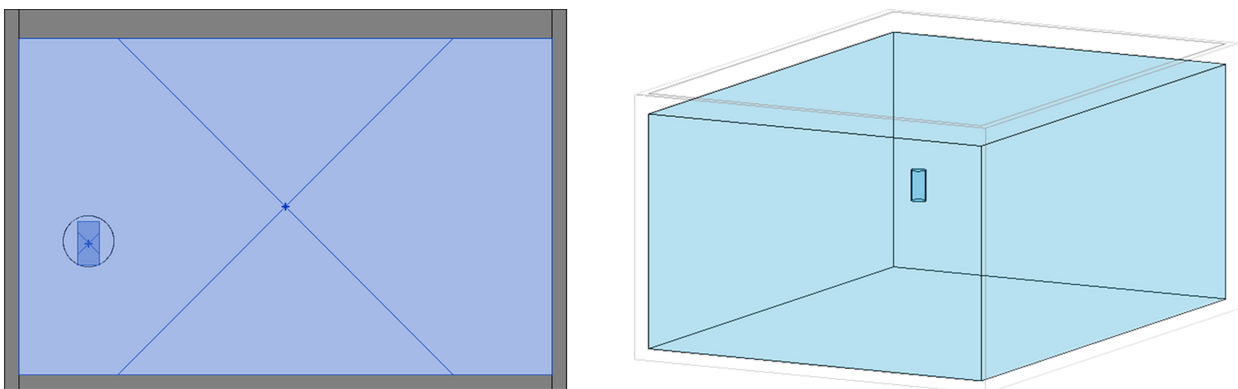


Fig. 3-17 - Section view and gbXML 3D view of the Sensor room

### 3.3.2.2.2. "GBXML EXPORT" MANAGEMENT

The volume computation for a space is based on its room-bounding components. By default, Revit does not compute room volumes in this way but computes them calculating the product of the area of its base and the height of the space, obtaining approximative results especially in presence of irregular or sloping surfaces.

Areas and volumes computation

In order to constrain Revit to do a more meticulous computation, we have switch on "Area and Volumes" in the Volume Computations panel under the Computations tab of the Area and Volume Computations dialog before exporting the model.

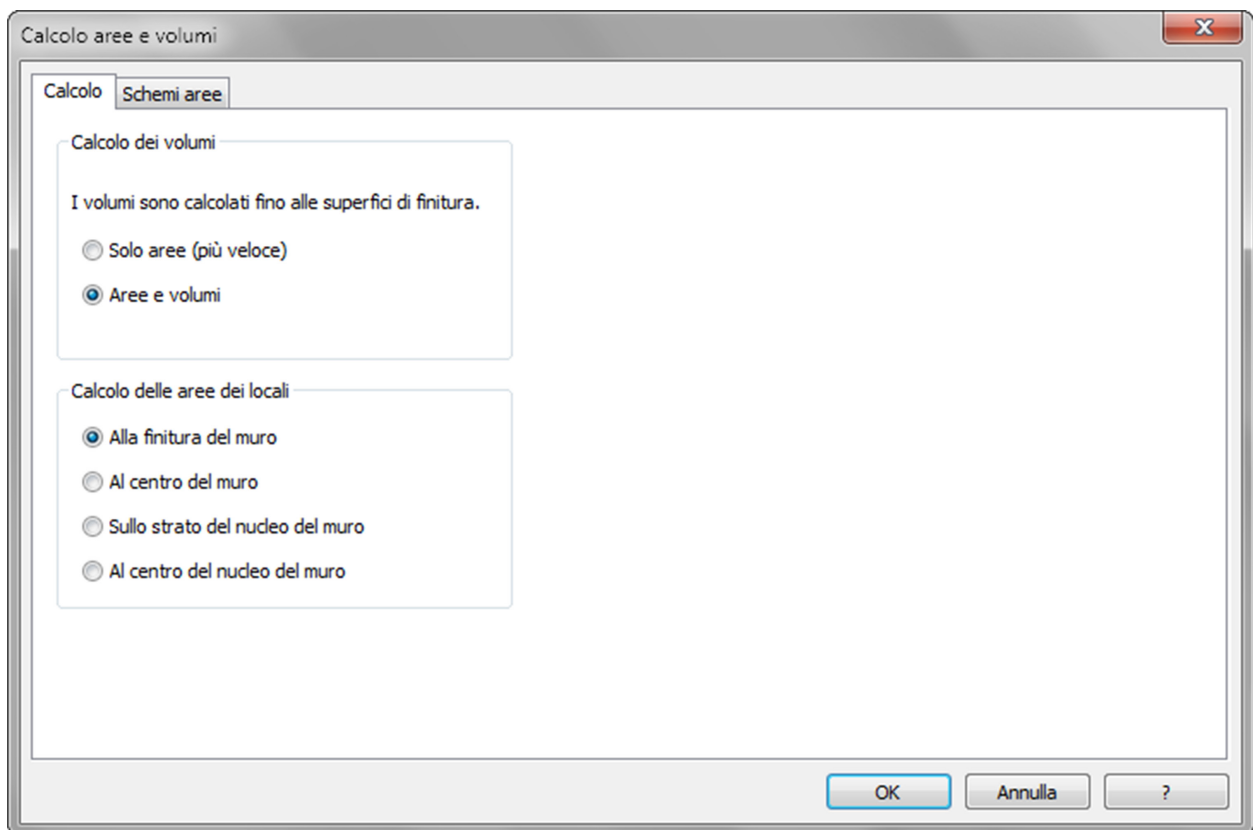


Fig. 3-18 - Areas and volumes computing setting window

Once made this, we can start personalizing the energy settings of the model. To do this it's necessary to go under the "Analyse" tab and click over the "Energy settings" icon. The window represented in the image below will appear.

Energy settings management

Parametro	Valore
<b>Comune</b>	
Tipo di edificio	Scuola o università
Posizione	Viale Piave, 10036 Settimo Torinese T
Piano del terreno	Plinti e Cordoli
<b>Modello dettagliato</b>	
Categoria di esportazione	Locali
Complessità esportazione	Complessa
Includi proprietà termiche	<input checked="" type="checkbox"/>
Fase del progetto	Stato di Fatto
Tolleranza vani ridotti	0.0000
Involuppo dell'edificio	Usa parametro Funzione
Dimensioni cella griglia analitica	0.9144
<b>Modello energetico</b>	
Modalità di analisi	Usa elementi di costruzione
Risoluzione vano analitico	0.2500
Risoluzione superficie analitica	0.0800
Offset nucleo	3600.0000
Crea zone	<input checked="" type="checkbox"/>
Costruzioni concettuali	Modifica...
Percentuale superficie vetrata	40%
Altezza avanzata	750.0000

Fig. 3-19 - Energy settings window

The first property set, "Common", gives the possibility to assign a building typology to the Revit model, to specify the geographical position of the project (if it wasn't already done) and to declare which project's level is the ground floor. The second one, the most important, allows to manage the exportation in all its aspects. We can in fact specify the following fields:

- Export Category: this option determines whether rooms or spaces are exported. There's no actual difference in results between the two possibility but, since in our project only the Room entities have been managed, the approach chosen is the first one;
- Export Complexity: this data specifies the level of detail provided



when generating gbXML data for openings, and whether shading surface information is exported. The possibilities range between simple options, in which curtain walls and curtain systems are exported as a single opening (without individual panels), and complex options, permitting to curtain walls and curtain systems to be exported as multiple openings, panel by panel;

- Include Thermal Properties: if Export Category is set to rooms, we've select this option to export thermal properties to gbXML. Note that if we don't select this field the Construction element and his child elements will not be exported;
- Sliver Space Tolerance: this field allows to specify a tolerance value for sliver spaces. All areas that are within the sliver space tolerance are considered sliver space.

### 3.3.2.3. GBXML MODEL ANALYSIS

#### 3.3.2.3.1. GEOMETRIES MONITORING

Before completing the gbXML exportation, it may be necessary to check for possible problems that might affect the success of the transition process and to make sure that the model is correctly configured for export from a geometrical point of view.

To do this we can use the 3D viewer offered by the default gbXML exporter included in Revit. We can open it by clicking on the Revit's R "file" button, then on "Export" and finally on "gbXML".

On the right we can manage again the energy settings we already set; once we've eventually modified these fields we can click on the "Detail" tab. At this point we have the possibility to choose between to level of detail to check the geometries:

- Rooms;
- Analytical surfaces.

The first possibility gives us the chance to navigate level by level in order to control the proper dispositions and extensions of the rooms. This tool offers also some visualization features like the "Highlight" command to make easier the navigation and visualization in the 3D viewer visible on the left.

But most important of all, clicking on the warning button it's possible to read a problem's review eventually connected to a non correctly defined room in the model. In this case we have to fix all the problems one by one, until all the warning icon disappears.

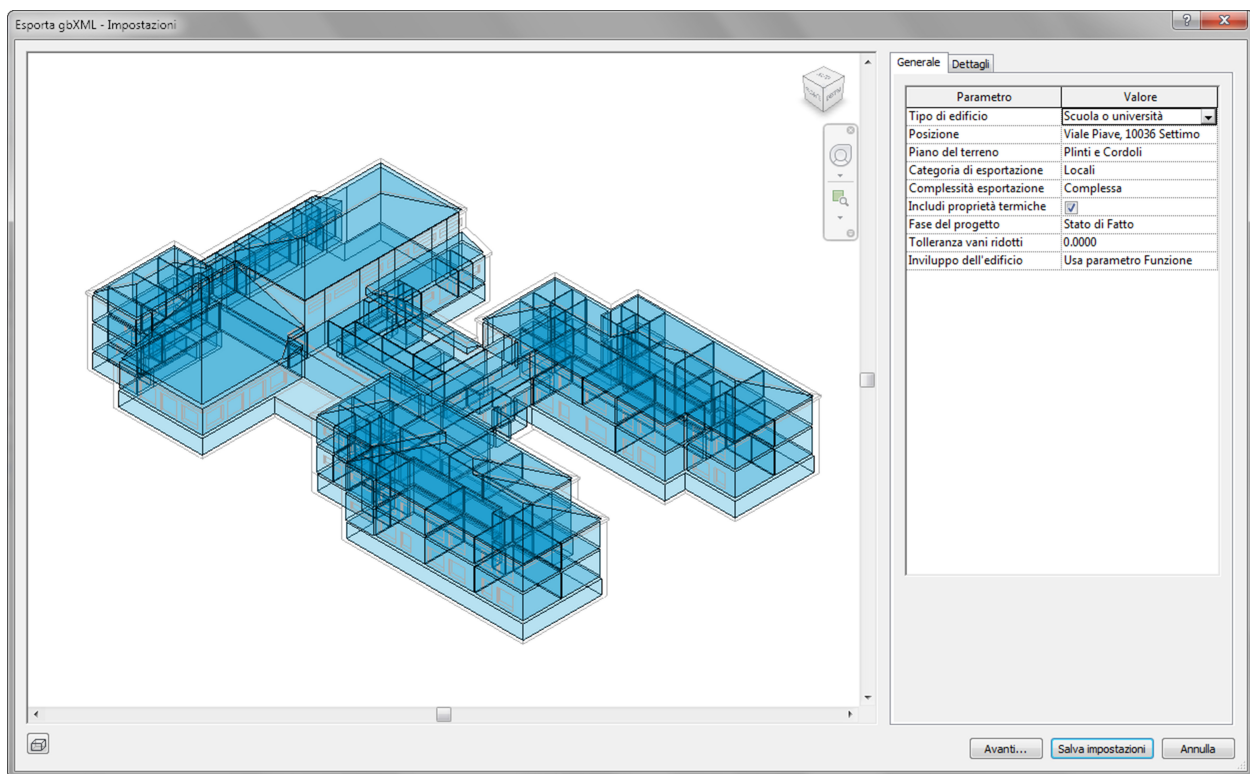


Fig. 3-20 - gbXML export window (Room detail)

Once we have checked that all the rooms are correctly placed and defined, we can analyse the model by a "Analytical surface". In this way the 3D view will generate separate analytical surfaces with different colour, to indicate the different categories they belong (Wall, Floor, Roof, Opening, etc.). This step is necessary to control if all the surfaces will be exported correctly and if there are undesired categories we don't want to be exported, such as Air.

The procedure proceed with the optimization of the model's geometry, following the tips exposed in "Model's geometries optimization" until all the surfaces are correctly disposed for the exportation.

Once the model geometries are checked, we can proceed clicking on the "Next..." button and complete the gbXML exportation, with the creation of a XML file.

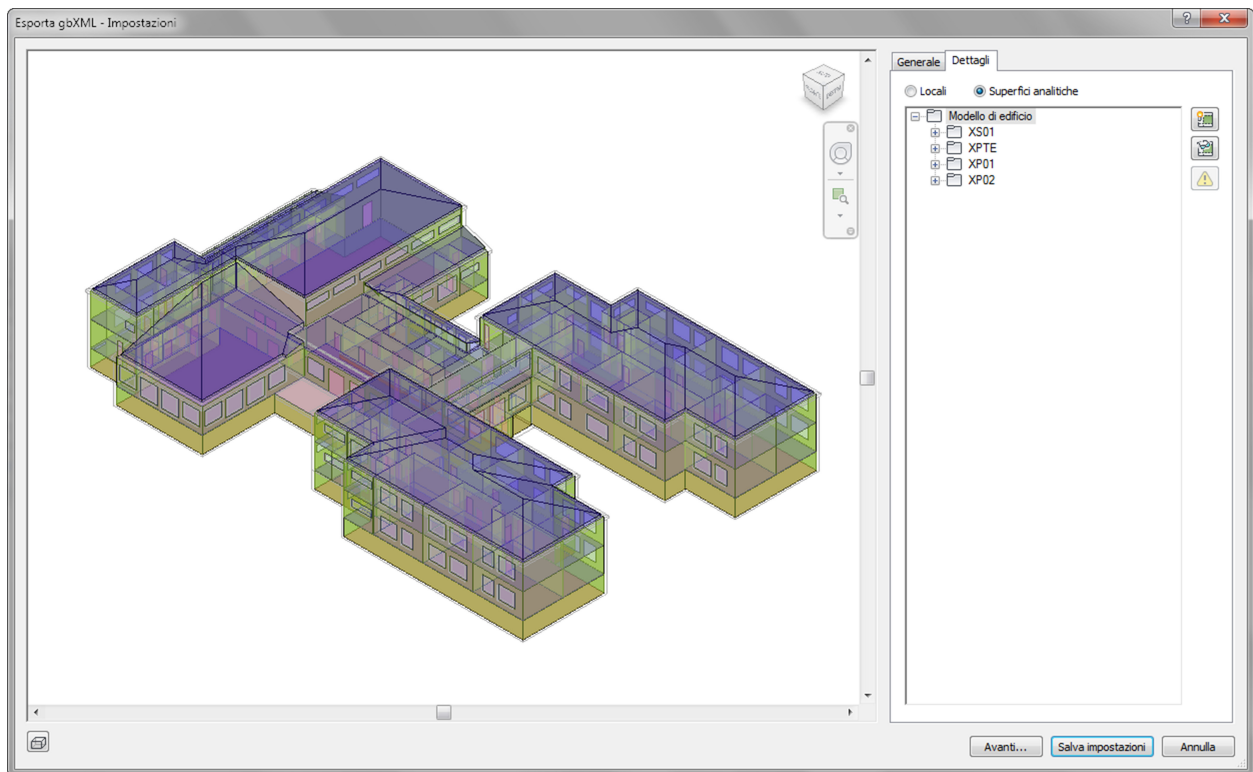


Fig. 3-21 - gbXML export window (Analytical surfaces detail)

### 3.3.2.3.2. PROPERTIES MONITORING WITH XML EDITOR

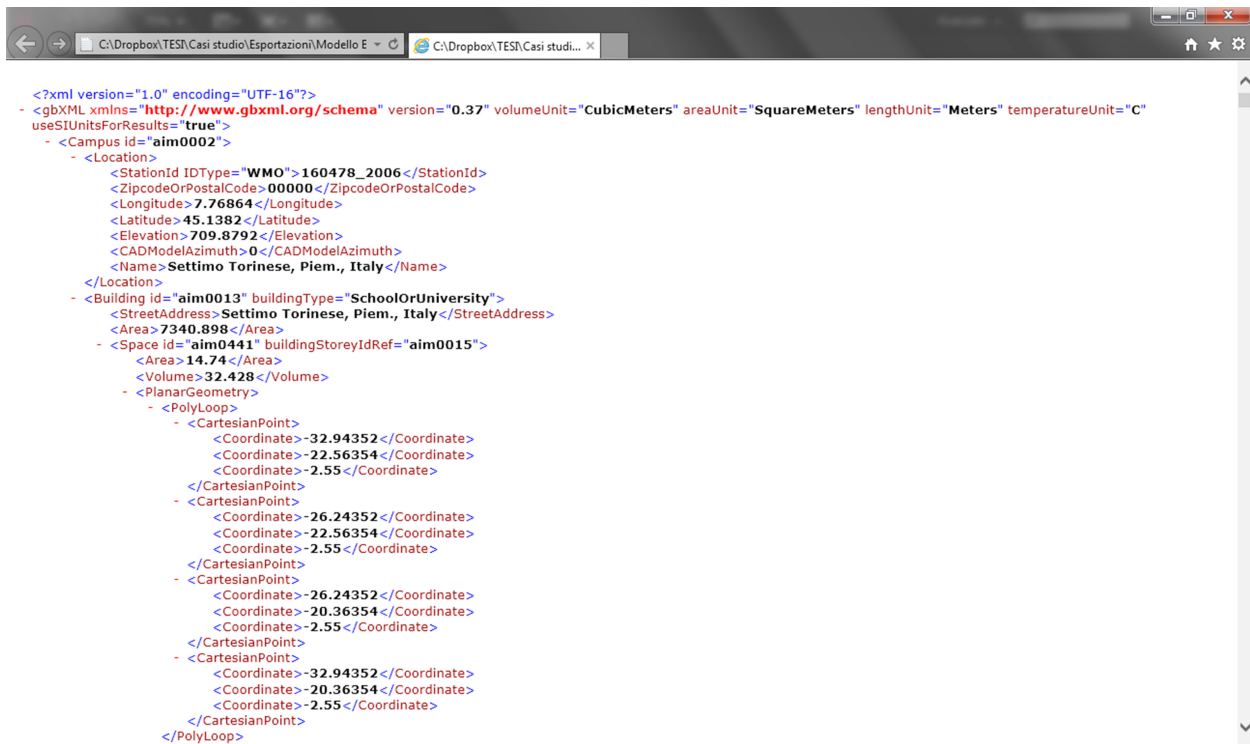
At this point we can proceed analysing the data structure of the gbXML model. To do this is necessary to open the XML file as a text document and this operation simply need a normal text reader software, but into this multitude of applications, we selected XML Editor, which it uses the Internet Explorer interface to open the XML editor in a interactive way, as visible in the image below.

This is due to the fact that XML Editor permits a better navigation and comprehension of the intricate nested structure of the document, thanks to his capability to open and close parent elements by clicking on the "-" and "+" symbols present on the left side of them. In this way is possible to parse the file step by step, going from the superior levels of detail to the most particular.

Another simple but important tool offered by this program is the "Search" feature, which allows to control if a particular element has been exported from Revit to the gbXML model or above all to check the connection between the different elements, using as investigation key the ID attributes that stand among them.

Most elements have in fact an ID univocal attribute contained into commas and that follows a specific encoding: "aimxxxx". At the place of the "x" characters it will be present a progressive number. This attribute allows the element to communicate between them thanks to specific IdRef attributes which call a

specific ID attribute in the model.



```
<?xml version="1.0" encoding="UTF-16"?>
- <gbXML xmlns="http://www.gbxml.org/schema" version="0.37" volumeUnit="CubicMeters" areaUnit="SquareMeters" lengthUnit="Meters" temperatureUnit="C"
  useSIUnitsForResults="true">
  - <Campus id="aim0002">
    - <Location>
      <StationId IDType="WMO">160478_2006</StationId>
      <ZipcodeOrPostalCode>00000</ZipcodeOrPostalCode>
      <Longitude>7.76864</Longitude>
      <Latitude>45.1382</Latitude>
      <Elevation>709.8792</Elevation>
      <CADModelAzimuth>0</CADModelAzimuth>
      <Name>Settimo Torinese, Piem., Italy</Name>
    </Location>
  - <Building id="aim0013" buildingType="SchoolOrUniversity">
    <StreetAddress>Settimo Torinese, Piem., Italy</StreetAddress>
    <Area>7340.898</Area>
  - <Space id="aim0441" buildingStoreyIdRef="aim0015">
    <Area>14.74</Area>
    <Volume>32.428</Volume>
    - <PlanarGeometry>
      - <PolyLoop>
        - <CartesianPoint>
          <Coordinate>-32.94352</Coordinate>
          <Coordinate>-22.56354</Coordinate>
          <Coordinate>-2.55</Coordinate>
        </CartesianPoint>
        - <CartesianPoint>
          <Coordinate>-26.24352</Coordinate>
          <Coordinate>-22.56354</Coordinate>
          <Coordinate>-2.55</Coordinate>
        </CartesianPoint>
        - <CartesianPoint>
          <Coordinate>-26.24352</Coordinate>
          <Coordinate>-20.36354</Coordinate>
          <Coordinate>-2.55</Coordinate>
        </CartesianPoint>
        - <CartesianPoint>
          <Coordinate>-32.94352</Coordinate>
          <Coordinate>-20.36354</Coordinate>
          <Coordinate>-2.55</Coordinate>
        </CartesianPoint>
      </PolyLoop>
    </PlanarGeometry>
  </Space>
</Building>
</Campus>
```

Fig. 3–22 - XML editor interface view

### 3.3.3. RVT - IFC EXPORTATION

#### 3.3.3.1. IFC FORMAT

##### 3.3.3.1.1. INTRODUCTION

###### MAIN FEATURES

IFC stands for “Industry Foundation Classes”, the set of internationally standardized object definitions for use in the Construction Industry, developed by the International Alliance for Interoperability (IAI).

The development of the IFC is strictly connected to the evolution of the Building Information Modeling universe.

IFC defines how informations should be provided/stored for all stages of a building projects lifecycle, from “very little” information to “everything”.

It is a neutral and open file format that is not controlled by a single vendor or supplier group. This is an object-based file format with a data model developed by the International Alliance for Interoperability (IAI) to facilitate interoperability between architecture, engineering and building industry disciplines.



Fig. 3–23 - IFC logo

###### HISTORY

The birth of the IFC is due to the necessity of creating a totally open standard, in order to face many problems that afflicted the world of the constructions. First of all it was necessary to improve the automation and the interoperability between the several softwares used in the design process and that were having a great development in the 90s.

The IFC initiative began when Autodesk started a consortium with the aim to involve several companies, active in the AEC field, in the development of a set of C++ classes that could support integrated application development. Since its debut in 1994, 20 US companies joined the consortium, that initially took the name of Industry Alliance for Interoperability and then that of International Alliance for Interoperability (IAI) in 1997. This alliance was born as a non-profit organization, with the goal of creating the IFC standard, a neutral and open specification that is not controlled by a single vendor or group of vendors. Every member of the IAI is very important for the definition of the standards by sharing their own experience and providing the informations that's needed to decide whether or not is becoming a standard.

There have been five principal releases of the IFC over since 1996; these had been IFC1, IFC1.5.1, IFC2.0, IFC2x, IFC2x2 and IFC2x3. Since the release of IFC2x

(published in 2000), each release has left the core (or ‘platform’) of the IFC specification unchanged, and has added to it. This platform guarantee has meant that most vendors have had no difficulty in upgrading their application to IFC2x2, IFC2x3 and hopefully in future to IFC2x4. Since the publication of IFC2x new releases were mainly driven by adding new concepts to the IFC specification in order to capture more exchange use cases and to improve existing definitions reflecting the lessons learnt from implementation and usage. Since IFC2x a new major release has been published every 3 years.

### 3.3.3.1.2. SYNTAX AND STRUCTURE

#### STEP AND EXPRESS INHERITANCE

The IFC specification is written using the EXPRESS data definition language, defined as in “ISO 10303-11: Industrial automation system and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual”. It has the advantage of being compact and well suited to include data validation rules within the data specification. The IFC exchange file structure (the syntax of the IFC data file with suffix “.ifc”) is the so called “STEP physical file” format (or “STEP- File”), defined as ISO 10303-21. It is an ASCII file format used to exchange IFC between different applications.

#### Standard for the Exchange of Product model data

STEP (Standard for the Exchange of Product model data) is a standard containing a set of rules for integration, representation and data-sharing and it is formalized by the specific ISO 10303. From a technical point of view, one of the advantages of STEP is its ability in supporting several protocols within a whole structure; every protocol contains a diagram, which describes the operations needed to pursue the target, and a model of requirements, that shows the infos required for that activities. Today the main advantage of STEP is its capacity of sharing project data as solid models; it has opened the path to the 3D data sharing that evolved in the definition for example of the IFC standard.

#### EXPRESS data modelling language

EXPRESS is a standard data modelling language for product data and, as above-mentioned, it is defined within the STEP specification, in ISO 10303 -11. It’s fundamental, first of all, specify what are data models: they define data objects and relationships among data objects for a certain domain of interest.

In light of this, we can give a proper definition of what is EXPRESS language: it provides for the modeling of data and data relationships with a very general and powerful inheritance mechanism and it includes a full procedural programming language which is used to specify constraints on data instances. The mechanisms it provides for defining the types of object and their properties, that will be used

in a given data set, and the constraints to which those objects should conform, are far richer than other exchange formats.

EXPRESS is therefore a language designed specifically to represent aspects of product data through schemas and constraints; it is not a programming language, although it was influenced by several programming languages. EXPRESS data schemas have two most common representations: a lexical and a graphical. The lexical form is stored in ASCII files, similar to many programming languages, and so looks like a plain text files. The language elements are formed into a stream of text, divided into physical lines, using a limited character set and defined special-purpose syntax. A physical line is any number (including zero) of characters ended by a newline.

The graphical representation of EXPRESS is called EXPRESS-G. It is a graphical modelling notation developed within STEP and used for IFC definition. It is used to identify classes, the data attributes of classes and the relationships that exist between classes and consists of semantically defined symbols comprised primarily of boxes, lines and small circular arrowheads. EXPRESS-G is directly related to the EXPRESS data definition language and everything is drawn in EXPRESS-G can be defined in EXPRESS; however, not everything that can be defined in EXPRESS can be drawn in EXPRESS-G. One of the advantages of using EXPRESS-G instead of EXPRESS is that the structure of a data model can be explained in a more understandable manner.

EXPRESS-G

## SYNTAX

The most generic element of the EXPRESS schema is declared using the SCHEMA keyword; within a SCHEMA various datatypes can be defined together with structural constraints and algorithmic rules. A schema is a collection of entities (or classes), attributes, and relationships between entities. It defines the patterns or templates by which populations of these entities and relationship shall be represented. Such a schema is often called a Product (Data) Model (as opposed to a populated data model). A model is a population of a schema, following the patterns, templates and constraints stipulated by the schema. It contains the actual instances of the entities (or classes). Such a model is often called a populated data model, a project data model, a building information model (if content is construction industry specific. An IFC exchange file represents a building (information) model.

Datatypes can belong to different categories, related into a hierarchical structure. In the following lines are shown these different datatypes categories, combined with the respective EXPRESS-G representation. This specification is useful in order to understand correctly the graphical representation reported during the study

Datatypes

and the analysis of the different IFC classes and the definition of the relationship between them.

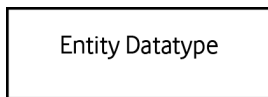


Fig. 3–24 - EXPRESS-G representation of Entity Datatype

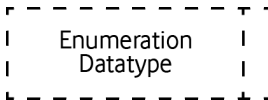


Fig. 3–25 - EXPRESS-G representation of Enumeration Datatype

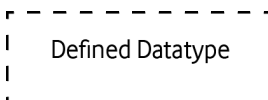


Fig. 3–26 - EXPRESS-G representation of Defined Datatype

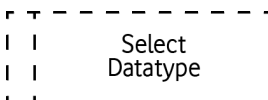


Fig. 3–27 - EXPRESS-G representation of Select Datatype

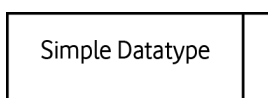


Fig. 3–28 - EXPRESS-G representation of Simple Datatype

- **Entity data type:** This is the most important datatype in EXPRESS. Entity datatypes are what we are mostly interested for and are also known as CLASSES. They can be related in two ways, in a sub-supertype tree and/or by attributes.

- **Enumeration data type:** Enumeration values are simple strings such as red, green, and blue for an rgb-enumeration. An enumerated data type provides for a range of possible values which the attribute may have described in an enumeration list. The attribute may only take one value from the possible range. It is shown as a rectangular box with dashed lines and a double vertical bar to the right. The name given to the enumeration is written in the box.

- **Defined data type:** This further specializes other datatypes (for example it define a datatype positive that is of type integer with a value > 0). A defined data type is used to take the place of a simple data type and is used to make the meaning of the model clearer. An organization may have a description which could take the form of a simple STRING data type. However, it might be more appropriate to make a data type called 'text' which could be used for the description. A defined data type is shown in EXPRESS-G as a rectangular box using dashed lines and the name given to the type written in the box.

- **Select data type:** Selects define a choice or an alternative between different options. Most commonly used are selects between different entity types. More rare are selects that include defined types. In the case that an enumeration type is declared extensible, it can be extended in other schemas. A select data type enables the choice of which direction to follow in the model; in effect, select the class to be used for a purpose. In this sense, it is similar to a supertype/subtype and its construction in EXPRESS-G looks similar. A select is shown in EXPRESS-G as a rectangular box using dashed lines with a double vertical bar at the left hand end and the name given to the type written in the box.

- **Simple data type:** is the atomic parts of EXPRESS and EXPRESS-G; that is, it cannot be subdivided into anything smaller. A simple data type is shown as a solid rectangular box with a double vertical line at the right hand side of the box. The actual name of the data type is enclosed within the box.

- **String:** This is the most often used simple type. EXPRESS strings can be of any length and can contain any character (ISO 10646/Unicode).



- Binary: This data type is only very rarely used. It covers a number of bits (not bytes). For some implementations the size is limited to 32 bit.
  - Logical: Similar to the boolean datatype a logical has the possible values TRUE and FALSE and in addition UNKNOWN.
  - Boolean: With the boolean values TRUE and FALSE.
  - Number: The number data type is a supertype of both, integer and real. Most implementations take uses a double type to represent a real\_type, even if the actual value is an integer.
  - Integer: EXPRESS integers can have in principle any length, but most implementations restricted them to a signed 32 bit value.
  - Real: Ideally an EXPRESS real value is unlimited in accuracy and size. But in practise a real value is represented by a floating point value of type double.
- **Aggregation data type:** The possible kinds of aggregation types are SET, BAG, LIST and ARRAY. While SET and BAG are unordered, LIST and ARRAY are ordered. A BAG may contain a particular value more than once, this is not allowed for SET. An ARRAY is the only aggregate that may contain unset members. This is not possible for SET, LIST, BAG. The members of an aggregate may be of any other data type. SET and LIST, finally, are the most used in the EXPRESS language.
    - ARRAY a fixed size collection of things with order represented as A[1:?].
    - BAG a collection of things with no order and allowed duplication represented as B[1:?].
    - LIST a collection of things with order and no duplication represented as L[1:?].
    - SET a collection of things with no order and no duplication represented as S[1:?].

The first character in square parentheses in an aggregation is the minimum possible value. The second character is the maximum possible value and may be either a number or the “?” character which means “indeterminate”.

For what involves the relationship between datas, there are many things to say. First of all it's necessary to specify that everything's related to a class is considered to be an attribute or data member. Attributes may be either mandatory or optional. Mandatory means that whenever is declared an instance of the class, a value of that attribute must be given. Optional means that a value may be given but that it is not necessary. Mandatory relations are represented through a solid line between class and attribute. On the other hand, optional relations are shown by a dashed line between class and attribute. The name of the relation is

Attributes

written above the line, without the use of the space character, and a circle shows the primary direction of the relation.

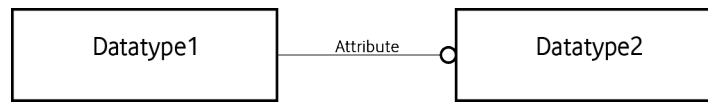


Fig. 3–29 - EXPRESS-G representation of Attribute

#### Supertype/subtype

Another important concept to notice is that the relations automatically create a supertype/subtype structure in which the attributes are subordinated to a inheritance mechanism. EXPRESS allows for the specification of entities as subtypes of other entities, where a subtype entity is a specialization of its supertype. This establishes an inheritance relationship between the entities in which the subtype inherits the properties of its supertype. Considering the case where there is a general specification for a class but that this is expanded by particular characteristics of subtypes, for the layered element, wall, floor and roof slab have already been indicated as subtypes. Each subtype has all the characteristics of the layered element acquired by inheritance. However, each subtype may have additional attributes. Supertype/subtype relations are a special form due to this inheritance capability, and are represented by a double thickness line.

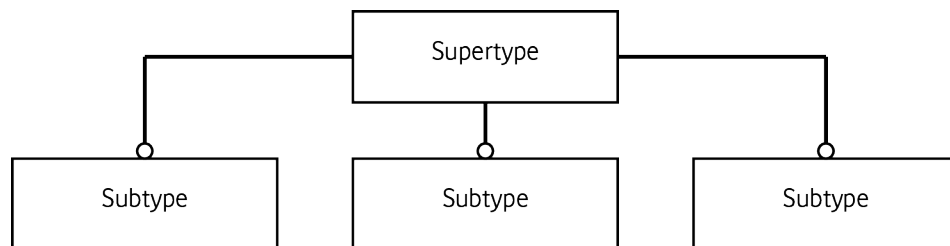


Fig. 3–30 - EXPRESS-G representation of Subtyping

#### Inverse relationship

The last concept to analyse is the inverse relationship. If another entity has established a relationship with the current entity by an explicit attribute, an inverse attribute may be used to describe that relationship in the context of the current entity. This inverse attribute may also be used to constrain the relationship further. An inverse attribute declaration also names an explicit attribute of the referencing entity.

#### IFC MODEL CLASSES BREAKDOWN

##### IfcRoot and fundamental classes

The IfcRoot is the most abstract and root class for all IFC entity definitions and the common supertype all all IFC entities. All entities that are subtypes of IfcRoot can be used independently. The IfcRoot class, in addition, assigns the globally unique ID, the ownership and the history information to the entity.

There are three fundamental classes in the IFC model, which are all derived from IfcRoot:

- IfcObjectDefinition: captures all object occurrences and object types;
- IfcPropertyDefinition: captures dynamically extensible property sets;
- IfcRelationship: captures relationships among objects.

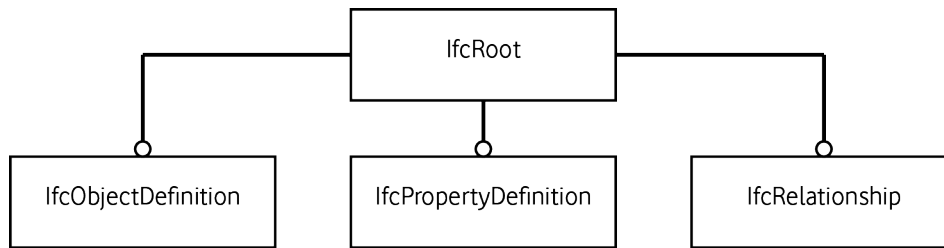


Fig. 3–31 - IfcRoot and fundamental classes

Once defined the main structure of the IFC model, it's fundamental to analyse the different occurrences, which are the fundamental part and also the concrete final output of the project matter of this thesis.

Objects, products and elements

All the occurrences and all their types are gathered within the IfcObjectDefinition class, that is the is the generalization of any semantically treated thing or process. Types are, in turn, gathered in the IfcObjectType class. Types are grouping of objects with a common definition of representation and properties.

IfcProduct is the root class for all physical objects and is subdivided into: spatial elements, physical elements, structural analysis items and other concepts. Products may have associated materials, shape representations, and placement in space.

Going into a more detailed decomposition of the IfcObjectDefinition hierarchical tree, we can also analyse the IfcElement class which include all components that make up an AEC product.

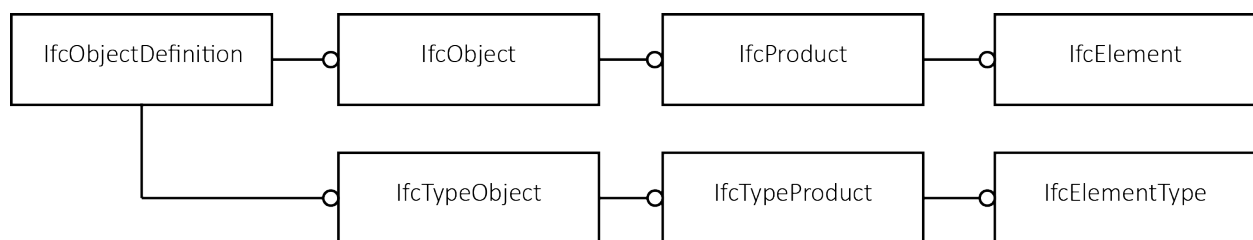


Fig. 3–32 - Objects, products and elements definition

The spatial structure of a building project can be described as “breakdown of the project model into manageable subsets according to spatial arrangements”. That means it is created by using the decomposition relationship as described in the image next, in which the involved instances are connected with the IfcRelAggregates subtype and create a hierarchical structure. The classes which compose the spatial structure are: IfcProject, IfcSite, IfcBuilding and IfcBuildingStorey. It is important to notice that only the IfcProject, IfcBuilding and IfcBuildingStorey are mandatory entities, while the IfcSite represent an optional level in the spacial structure. These classes all belongs to the IfcObject supertype.

Spatial structure and space elements

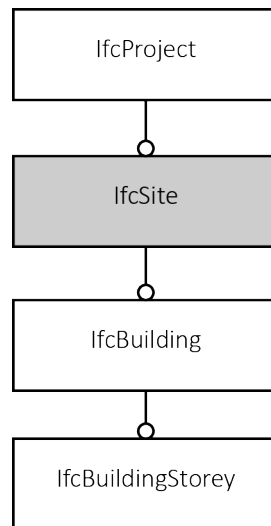


Fig. 3–33 - Mandatory and optional (in grey) levels of the spatial structure

Every IFC exchange file has inside the presence of one and only `IfcProject` instance. The `IfcProject`'s main purpose in an exchange structure is to provide the root instance and the context for all other information items included and to establish also the context for information to be exchanged or shared, and it may represent a construction project but does not have to.

It has a several functions in the IFC model: the first one is to establish the context of all representations within the project, the second one is to relate, as the uppermost container class, to all products, also the highest spatial container class instances (as `IfcSite` or `IfcBuilding`).

The `IfcSite` instance is used to give further information about the building site, such as geographical coordinates and address, and a geometrical representation of the surrounding terrain. The geometrical placement of the site, defined by the `IfcLocalPlacement`, shall be always relative to the spatial structure element, in which this site is included, or absolute, referring to the world coordinate system, as established by the geometric representation context of the project.

The building, represented by the `IfcBuilding` class, is used to issue additional information about the building itself. A building may range over several connected or disconnected buildings. A building can also be eventually decomposed in parts, where each part identifies a building section.

The `IfcBuildingStorey` instance is used to provide information about the building storeys, described as a horizontal aggregation of spaces that are vertically bound provided with an elevation information.

## GEOMETRIES

The first informations about the geometry and the spatial placement of the

element in the IFC model derives directly from the `IfcProject` instance, through the `IfcRepresentationContext` and his subtypes `IfcGeometricRepresentationContext` and `IfcGeometricRepresentationSubContext`. These classes define the context to which all the shape representations of products relate through their `IfcRepresentation` instance. In particular, in the `IfcRepresentationContext` we can find information about the World Coordinate System, the precision of the measurements and the true North direction.

The other information about the geometry and the position of the elements derives from the `IfcProduct` instance of that same product. Every object in the IFC file that derives from `IfcProduct`, in fact, can have geometric representations. Any object in IFC that has a geometric representation needs to have a value for the two attributes, inherited from `IfcProduct`: `ObjectPlacement` and `Representation`.

The `ObjectPlacement` attribute gives, through the `IfcObjectPlacement` information about the placement of the element in the spatial context, which can be of three different types:

Object placement

- Absolute: relative to the World Coordinate System;
- Relative: relative to the placement of another product;
- Constrained: relative to grid axes.

In particular, referring to the spatial structure of the IFC model previously analyzed, it's important to notice that only the `IfcSite` instance has to be placed absolutely (referring so only to the World Coordinate System declared within the `IfcProject`); `IfcBuilding`, on the contrary, shall be placed relatively to the `IfcSite` placement and in the same way `IfcBuildingStorey` instance is relative to the `IfcBuilding` container placement.

Regarding to the specific geometric elements contained in the IFC model (which can all relate to the generic class of `IfcElement`) we can individuate two different way of placing them in the spatial context relatively to:

- The local placement of its container (normally to the `IfcBuildingStorey` to which is associated;
- The local placement of the `IfcElement` to which is constrained.

The placement of the `IfcProduct` is ruled by the `IfcLocalPlacement` (for the first case) and the `IfcGridPlacement` (for the second one) subtypes. In the first case the `RelativePlacement` attribute leads to the `IfcAxis2Placement2D` and the `IfcAxis2Placement3D` Select Datatype, that gives the geometric placement and the axis direction for the concerned object. In the second case the definition of the placement is relative to the intersection of two grid lines (`IfcGridAxis`) within the same grid.

```
/* Definition of the world coordinate system */  
#1=IFCGEOMETRICREPRESENTATIONCONTEXT($, '3Dmodel', 3, 1.0E-05, #2, $);
```

```

#2=IFCAXIS2PLACEMENT3D(#3, $, $);
#3=IFCCARTESIANPOINT((0.0, 0.0, 0.0));
/* Definition of the local absolute coordinate systems */
#4=IFCLOCALPLACEMENT($, #7);
#7=IFCAXIS2PLACEMENT3D(#10, $, $);
#10=IFCCARTESIANPOINT((3.8, 1.4, 0.0));
#5=IFCLOCALPLACEMENT($, #8);
#8=IFCAXIS2PLACEMENT3D(#11, #16, #17);
#11=IFCCARTESIANPOINT((1.2, 1.4, 0.0));
#16=IFCDIRECTION((0.0, 0.0, 1.0));
#17=IFCDIRECTION((0.96592, 0.25881, 0.0));

```

Example of IfcLocalPlacement schema

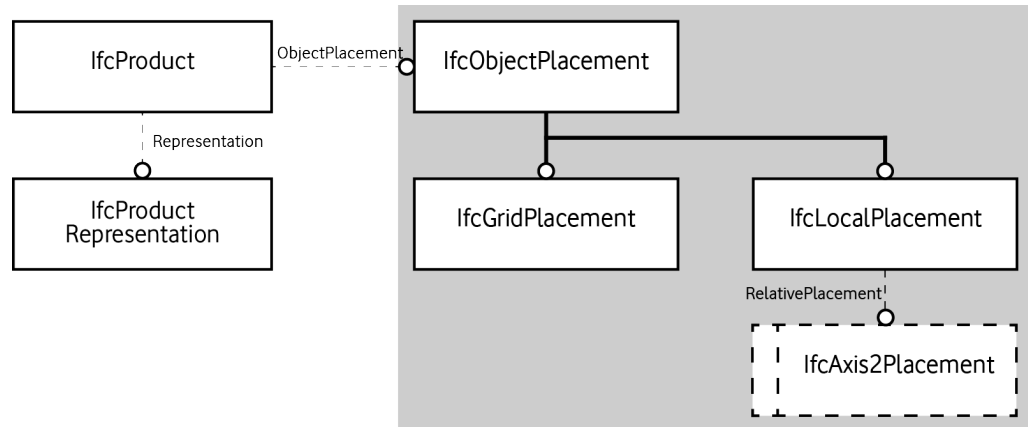


Fig. 3–34 - Definition of IfcObjectPlacement

## Representation

The Representation attribute leads first of all to the IfcProductRepresentation subtype, which is the container for all representations of the same product. It's possible in fact for a product to have different graphical representations, all grouped in the IfcProductRepresentation box. A product can have one or various geometric representations, and a single geometric representation can be shared among multiple products.

Each IfcProductRepresentation has to have at least one IfcRepresentation subtype, which defines the general concept of representing product properties through the use of two attributes: RepresentationType, used to declare the geometric model used for the shape representation, and RepresentationIdentifier, which denotes the part of the representation captured by the IfcShapeRepresentation. The IfcShapeRepresentation represents the concept of a particular geometric representation of a product or a product component within a specific geometric representation context. The most used representation types are listed below:

- Curve2D: 2 dimensional curves;
- GeometricSet: points, curves, surfaces (2 or 3 dimensional):
  - GeometricCurveSet: points, curves (2 or 3 dimensional);
  - Annotation2D: points, curves (2 or 3 dimensional), hatches and text (2 dimensional);
- SurfaceModel: face based and shell based surface model;

- SolidModel: including swept solid, Boolean results and Brep bodies  
more specific types are:
  - SweptSolid: swept area solids, by extrusion and revolution;
  - Brep: faceted Brep's with and without voids;
  - CSG: Boolean results of operations between solid models, half spaces and Boolean results;
  - Clipping: Boolean differences between swept area solids, half spaces and Boolean results;
  - AdvancedSweptSolid: swept area solids created by sweeping a profile along a directrix.

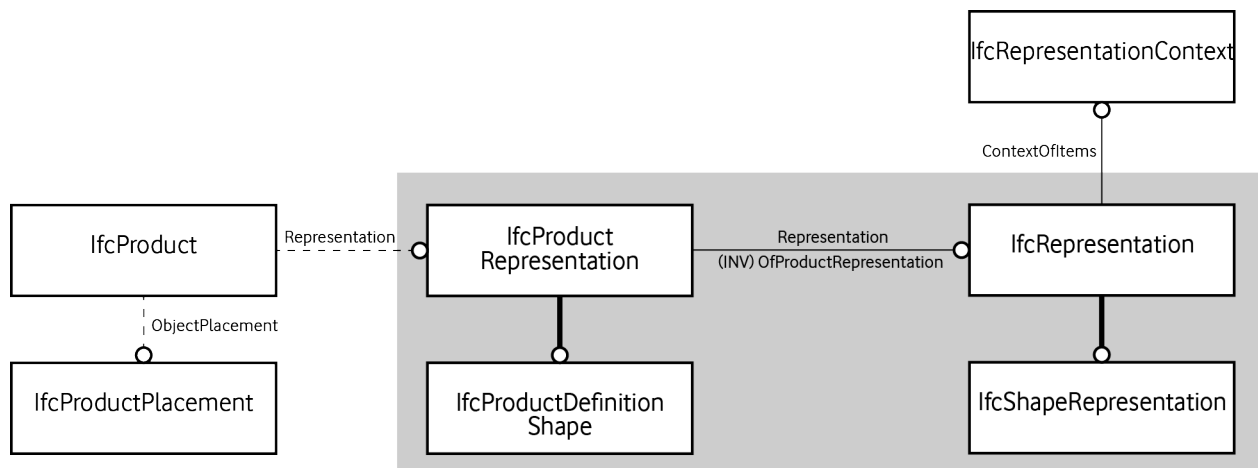


Fig. 3–35 - Definition of IfcProductRepresentation

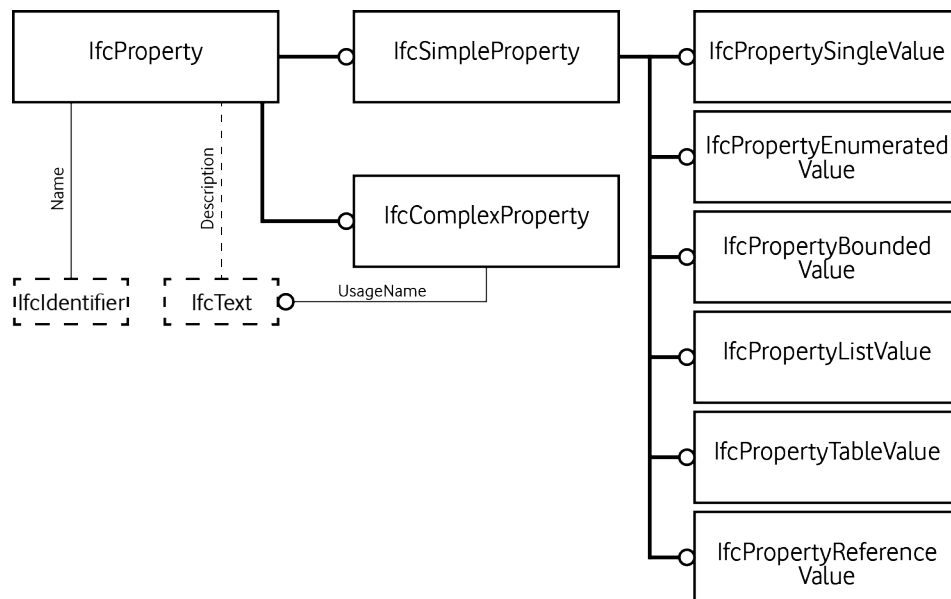
## PROPERTIES

The single properties can be distinguished between:

Properties

- Statically defined properties: they are declared through their own entity type and the meaning of the properties is defined by the name of the explicit attribute.
- Dynamically extendable properties: they define properties for which the IFC model only provides a kind of “meta model”, specified within the IfcProperty class. This means no entity definition of the properties exists within the IFC model. The declaration is done by assigning a significant string value to the “Name” attribute of the entity as defined in the entity IfcPropertySet and at each IfcProperty, referenced by the property set.

A Property of the second category must belong to one of the following IfcSimpleProperty, represented in the image below.



## Property Sets

Properties are stored in Property Sets and every Property Set exists only if it contains at least one Property. All Property Sets are a subtype of the `lfcPropertySetDefinition` class, which is an abstract supertype of property sets, including:

- Statically defined property sets: declared using their specific class name.
- Dynamically defined property sets: declared through the generic `lfcPropertySet` class;

IfcPropertySet is a container class that holds properties within a property tree, interpreted according to their name string attribute. As the name itself explains, this entity is a SET aggregation data type, that means the order of appearance of the attributes isn't important.

## Property Set Definition (PSD) Schema

Some Property Sets have predefined instructions on assigning significant name to their attributes, according to an explicit specification released by the IAI. That's because, since there are numerous alphanumeric attribute definitions depending on discipline, life-cycle stage, building regulation and region, there will never be a complete set of internationally standardized attributes. However, the IAI intent to standardize a basic set of properties, by releasing an external Property Set Definition Schema (PSD Schema) in which are defined a great number of Pset elements. These consists of a list of pre defined attributes and are identifiable by a specific and encoded name. The naming convention "Pset\_Xxx" applies to those property sets and shall be used as the value to the Name attribute.

Are listed below two examples of these particular Pset, which gather together



the properties of a kind of wall and one of door:

- Pset\_WallCommon: Reference; AcousticRating; FireRating; Combustible; SurfaceSpreadOfFlame; ThermalTransmittance; IsExternal; ExtendToStructure; LoadBearing; Compartmentation.
- Pset\_DoorCommon: Reference; FireRating; AcousticRating; SecurityRating; IsExternal; Infiltration; ThermalTransmittance ; GlazingAreaFraction; HandicapAccessible; FireExit; SelfClosing; SmokeStop.

The names and the related attributes of these Pset are reported in the IFC model respectively as the Property Sets' name and as the single property name.

```
#6341= IFCPROPERTYSET('Reference',$,IFCIDENTIFIER('Muro di base:INT10'),$);  
#6342= IFCPROPERTYSET('LoadBearing',$,IFCBOOLEAN(.F.),$);  
#6343= IFCPROPERTYSET('ExtendToStructure',$,IFCBOOLEAN(.T.),$);  
#6344= IFCPROPERTYSET('IsExternal',$,IFCBOOLEAN(.F.),$);  
#21110= IFCPROPERTYSET('ThermalTransmittance',$, IFCTHERMALTRANSMITTANCEMEASURE (4.09259259259259),$);  
#21111= IFCPROPERTYSET('0vqduHLff0k9GtsQSziBl',#41,'Pset_WallCommon',$,(#6341,#6342,#6343,#6344,#21110));
```

Example of Pset application exportation in the STP File

Property sets, defining a particular type of object, can be assigned in three different ways:

Properties assignments mechanisms

- Relating of an object type, for which a set of properties is defined. This is done though assigning an IfcTypeObject (through the IfcRelDefinesByType relationship) to a single or multiple object occurrences;
- Sharing a standard set of values defined in an IfcPropertySet (or any other subtype of IfcPropertySetDefinition) across multiple instances of that class (through the IfcRelDefinesByProperties relationship);
- Defining different property values within a private copy of the IfcPropertySet for each instance of that class.

The IfcTypeObject instance is attached to an object using the relationship class IfcRelDefinesByType. This allows for the object and the IfcTypeObject to exist independently and for the IfcTypeObject to be attached to the object when required. An advantage of using the relationship class is that the object does not contain any references to property set definitions if none are needed. Use of the relationship class also allows the IfcTypeObject to be attached to one or many objects. That is, many objects may share a single IfcTypeObject with its contained property set definitions if required.

Type Object attachment

Both, statically and dynamically defined property sets are attached to an object using the relationship class IfcRelDefinesByProperties. This allows

Property Set Definition attachment

for the object and the property definition to exist independently and for the `IfcPropertySetDefinition` to be attached to the object when required. An advantage of using the relationship class is that the object does not contain any references to property set definitions if none needed. Use of the relationship class also allows the property set definition to be assigned to one or many objects. That is, many objects may share a single property set definition with common values if required.

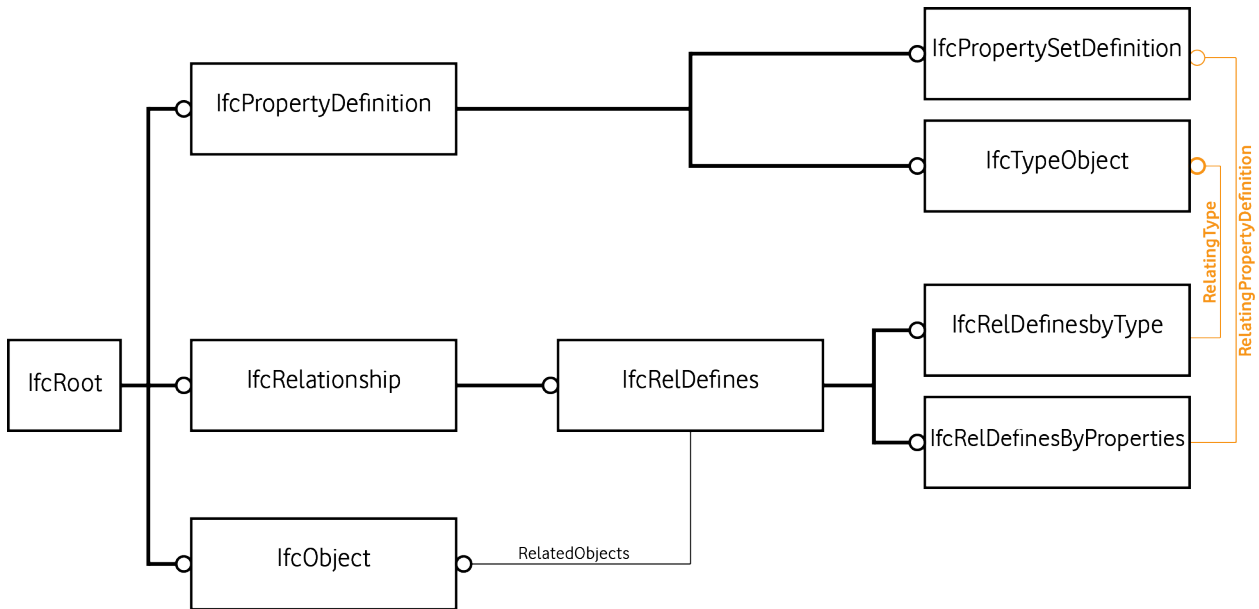


Fig. 3–37 - Properties attachment

#### Material properties association

The question about the material properties follows a separate path. Above all is fundamental to notice that the only elements in the IFC model that have this kind of information are those belonging to the `IfcElement` class (subtype of `IfcProduct`). Those elements are associated to the `IfcMaterialSelect` Select Datatype through the `IfcRelAssociatesMaterial` relationship. At this point, the `IfcElement` instance is redirected to one of the following classes:

- `IfcMaterial` (to be used in case of one single solid material);
- `IfcMaterialList` (for multiple material elements when precise structure is not specified);
- `IfcMaterialLayerSetUsage` (for layered elements, where the structure is specified).

An important consideration useful for the development of the present project is the exportation of the chromatic representation of the materials, that is made possible by the `IfcColourRGB` entity, which is associated directly to `IfcElement` geometry and, especially, his surface representation. This entity gives the coordinates of the RGB colour map (e.g.: the white colour must have coordinates 1,1,1).

### 3.3.3.2. PERSONALIZATION OF THE EXPORTATION: CREATING A STANDARD

#### 3.3.3.2.1. PRELIMINARY OPERATIONS ON REVIT

The exportation in the IFC format starts with a proper planning from the Revit model.

The first step in this preliminary procedure is the setting of a specific project view in which we are going to hide all the objects and occurrences we don't want to include in the file exportation. For this scope it has been preferred a 3D view, for the possibility of having a better control over the global visualization of the project, monitoring in the same time the object placed in all the different levels. This view has been called "IFC\_vista3D".

Views setting

Firstly it has been made a selection of the visible objects throughout the visibility filter, hiding the categories we don't want to see in the view, according to what's declared in "3.2. Planning phase". It's very important to admit the visualization of the masses since they have a key role in the final visualization.

Once we made this step, it may be necessary to do an additional filtering by selecting single instances in the view and hiding them by the "Hide in view" function, accessible by right-clicking on the element.

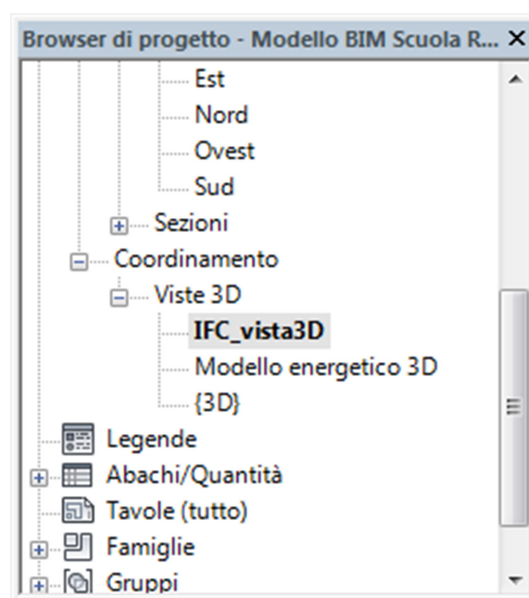


Fig. 3–38 - Revit's browser view of the 3D IFC-dedicated view

This procedure it's useful to take under control the elements visible and then exportable contained in the view, in order to check the absence of certain objects or the unnecessary presence of other ones. This involve the creation of a IFC model containing only useful elements, simplified and lightened for a easier

manageability.

### Schedules setting

The second step in the preliminary part of the IFC exportation is the creation of many project-dedicated schedule views in Revit. This choice is due to the necessity of having a better control over the properties exportation and a lightening of the informations load of the IFC model. These schedules have all the same codification, which includes in order:

- The "IFC\_abaco" identifier;
- The specification of the Level of Detail (LoD) for which that schedule it was made;
- The category of objects to which the schedule is dedicated.

This procedure is due to the necessity of easing the reading of the IFC file, making a preliminary selection of which parameters we want to export for each category of objects and a classification focused on declaring the LoD to which those attributes are dedicated.

The possibility of gathering in schedules just the attributes we want to export, gives the opportunity to pore only over the information useful for the exportation model, checking in this way if all the fields in the schedule are correctly filled. The exclusion of the unnecessary attributes is allowed by the correct setting of the IFC export parameters, as explained later in the following paragraphs.

```
#6348= IFCPROPERTY SINGLEVALUE('Famiglia',$,IFCLABEL('Muro di base: INT10'),$);  
#6349= IFCPROPERTY SINGLEVALUE('Tipo',$,IFCLABEL('Muro di base: INT10'),$);  
#6350= IFCPROPERTY SINGLEVALUE('Area',$,IFCAREAMEASURE(53.0250000000041),$);  
#6351= IFCPROPERTY SINGLEVALUE('Volume',$,IFCVOLUMEMEASURE(5.27487899999875),$);  
#6352= IFCPROPERTY SINGLEVALUE('Vincolo di base',$,IFCLABEL('Livello: XPTE'),$);  
#6353= IFCPROPERTY SINGLEVALUE('Vincolo parte superiore',$,IFCLABEL('Livello: XP01(-1)'),$);  
#6354= IFCPROPERTY SINGLEVALUE('Altezza non collegata',$,IFCLENGTHMEASURE(3.5),$);  
#6355= IFCPROPERTY SET('2rVmLy71PEzvENrR026qaD',#41,'IFC_abaco_LoD2_muri',  
$,(#6348,#6349,#6350,#6351,#6352,#6353,#6354));
```

Example of customized schedule exportation as IFCPropertySet

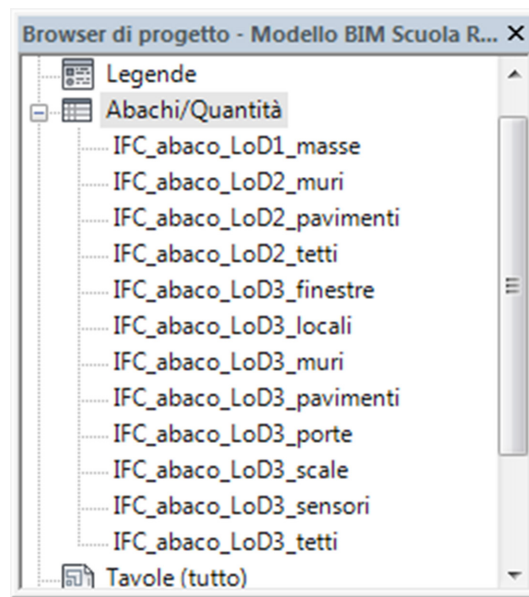


Fig. 3–39 - Revit's browser view of the list of schedules

### 3.3.3.2.2. “IFC EXPORT OPTIONS” MANAGEMENT

Once we have set correctly the view and the schedules we can start customizing the exportation parameters, starting from the creation of a IFC Mapping file. The file is a simple tabbed text file that contains three entries per line:

IFC Mapping file

- Revit category: declares the Revit family contained in the model;
- IFC class name: declares the class to which convert the Revit family;
- IFC type: declares the subcategory of the previously mentioned IFC class.

For standard building elements, an assigned class name appears in the IFC Class Name column. For building elements that do not have automatic mapping to IFC export classes, the value displayed in the IFC Class Name column is "Not Exported". If the values for a category or subcategory are blank, Revit will try to determine the appropriate category. If a match cannot be found for an object that has geometry, it is exported as a proxy object.

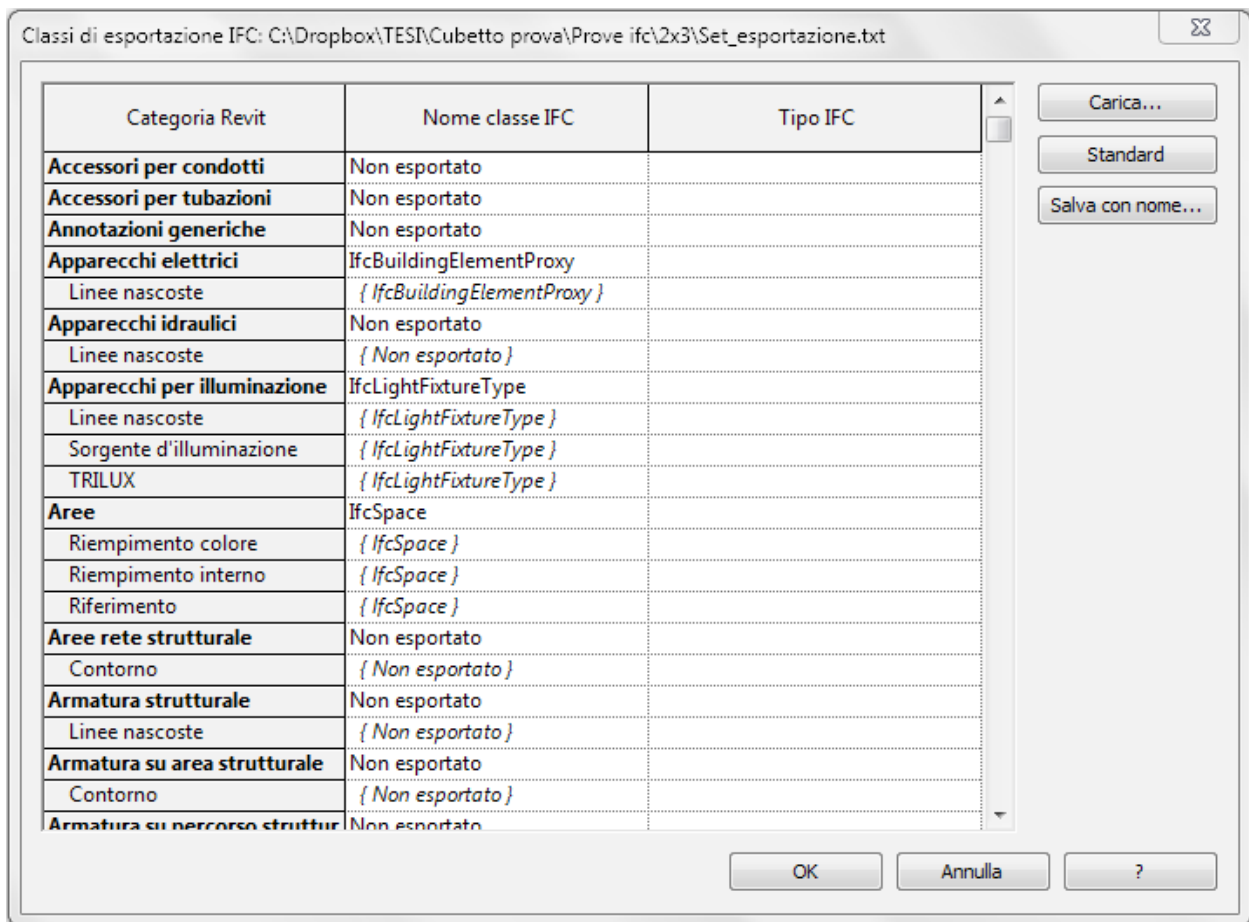


Fig. 3-40 - IFC Export Classes dialog panel

IFC Export Classes dialog panel

The first thing to do is so to map the generic family instances to IFC containers, making a selection of the element we want to export and assigning them to the most convenient IFC class, taking care about the list of the IFC classes supported by Revit. At the same time we can fill with "Not exported" if we don't want to export those Revit elements.

This procedure may seem redundant if we look at the procedures made in the creation of the "IFC\_vista3D" view, because we already did in that moment a selection of exportable and non exportable objects. But in that case the operation was made in order to have a graphical result of the geometries we want to convert.

In this case the IFC Export Classes dialog panel doesn't give a graphical representation of the final result, but it allows to make a greater step in the creation of a standardised exportation. In fact, once we have completed the assignment procedure, we can save all the changes into a new IFC Mapping file and this file will work as a fundamental document in this context, since it can be shared between different users and applied to multiple projects, which will be exported in the same way.

In this case all the changes have been saved in the "Set\_esportazione" IFC mapping file.



Set\_esportazione

Fig. 3-41 - IFC mapping file

### 3.3.3.2.3. "IFC EXPORT PARAMETERS" MANAGEMENT

Once we made all the association between Revit families and IFC containers, we can start customizing the exportation parameters. To do this it's necessary to make a selection of many features offered by the IFC Export plug-in software. In the interests of providing fuller information, we must notice that the Revit software itself has a IFC export feature, but it doesn't offer a great choice of personalization.

The plug-in, even if it was already installed with the standard Revit package, downloaded from the Autodesk site, it has been replaced with a newer version, more updated, since there were different features added in the newer one which are very useful for the scope of the project. The current version used is the IFC for Revit 2016 (v16.5).

The plug-in is accessible by clicking in the "R" home icon of Revit and then on "Export". Clicking on "IFC" the plug-in window, visible in the image below, opens automatically.

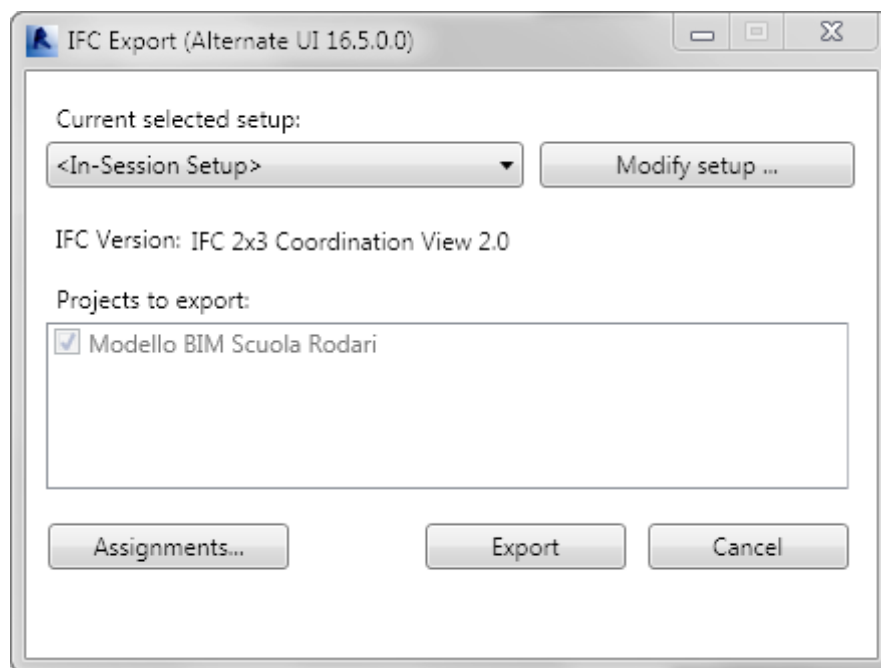


Fig. 3-42 - IFC Export main panel

In the left-upper part it's possible to chose between many standard setups. These built-in setups, when selected, involve a pre defined selection of the different options editable, and they cannot be modified or deleted, but they can be duplicated in order to create a modified version. The "In-Session Setup" is a modifiable setup which will not be saved between sessions but it admits a complete and free configuration of the options. We can add additional named setups using the "Setup Configuration" dialog box using the New or Duplicate options. These configurations will be added to the active document and saved

with the document. We can also rename and delete named setups from this dialog.

According to this, it is evident the potentiality of this tool in the perspective of the creation of a standardized exportation. For this reason we made some cautious choices in the editing of the different options that are visible in the images below. All these customised configurations are all gathered and saved in a project-dedicated new setup called "Setup\_TESI", in order to attach it to the Revit project and share it between different similar projects.

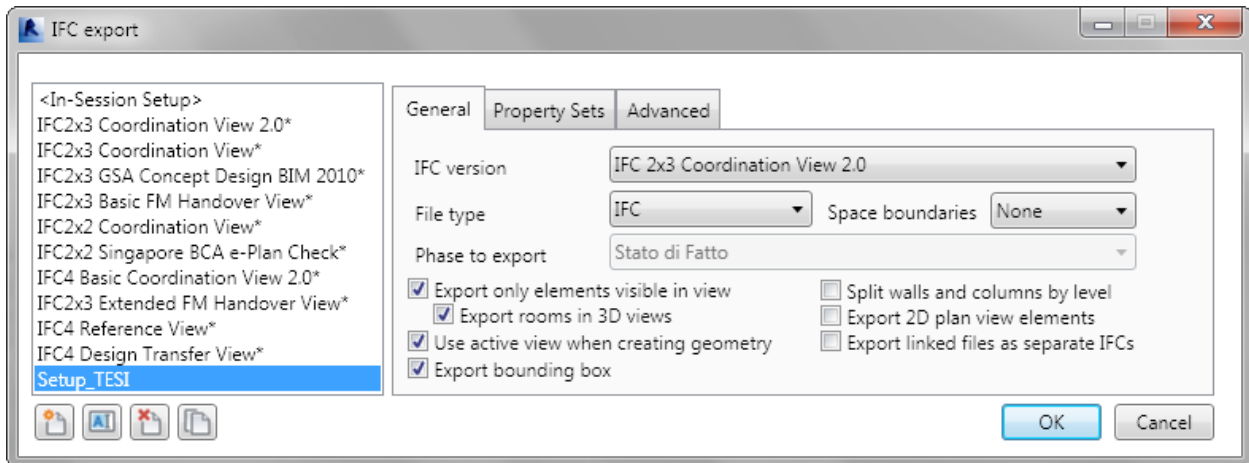


Fig. 3-43 - IFC export-setup panel, tab 1

The tab represented above is very important for the management of the geometries. It's important first of all to choose the correct IFC version and file type. The IFC standard selected is the "IFC 2x3 Coordination View 2.0", since it is the most accurate version of the IFC 2x3 native standard, while the file type selected is obviously "IFC", which corresponds to the creation of a IFC-SPF ("IFC Step File").

Its important to select the first box "Export only elements visible in view", and for this reason in also fundamental to open the IFC Export plug in only after we have opened and visualized the "IFC\_vista3D" view we've created as described in the previous paragraph.



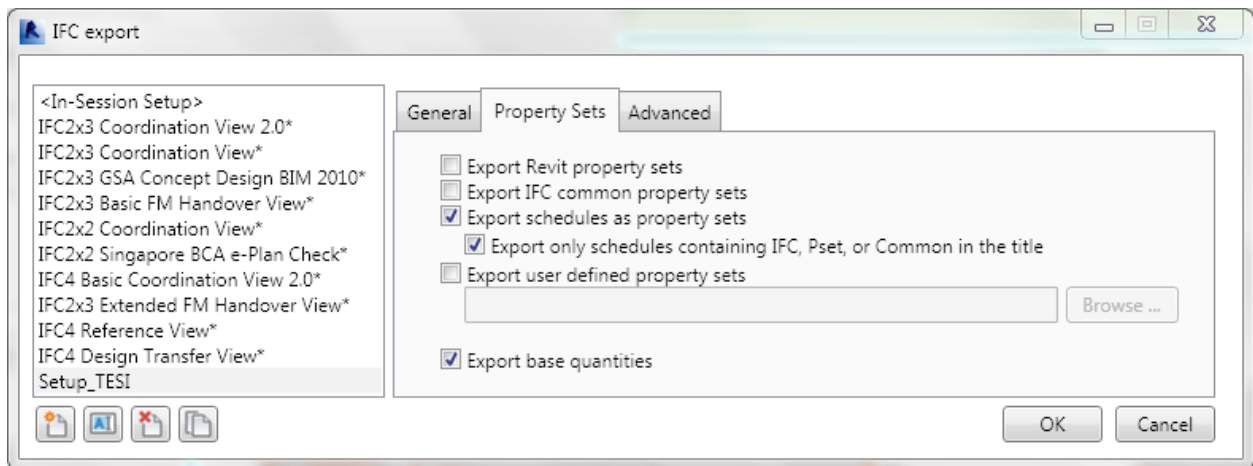


Fig. 3–44 - IFC export-setup panel, tab 2

This second tab is fundamental in order to manage the exportation of the informations included in the Revit model. In fact, since we made the preliminary operation of editing schedules described in the previous paragraph, we are interested in exporting only the informations included in these ones. For this reason we must check the "Export schedules as property sets" and the "Export only schedules containing IFC, PSet or Common in the title". By this way, the only information that will be exported are those contained in the schedule views created expressly for this scope and those described as "Base quantities". Base quantities are generated from model geometry to reflect actual physical quantity values, independent of measurement rules or methods.

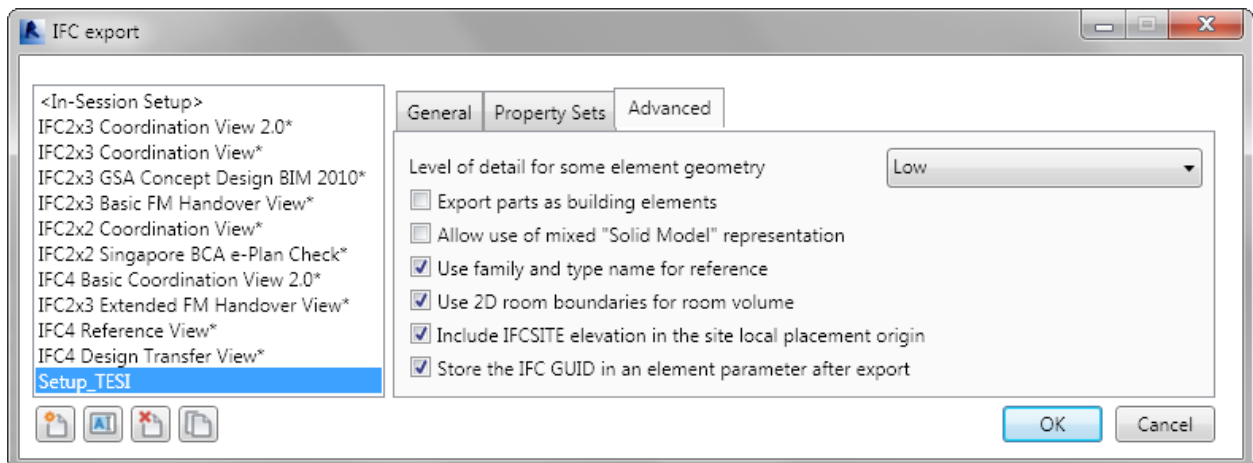


Fig. 3–45 - IFC export-setup panel, tab 3

Once we have created and saved the correct exportation setup, returning to the IFC Export main panel, we can see in the lower part the "Assignements..." Box. Clicking on it its possible to attach some fundamental informations to the IFC file. These informations are not internal to the IFC Model but are simply exported with it, attached as notes.

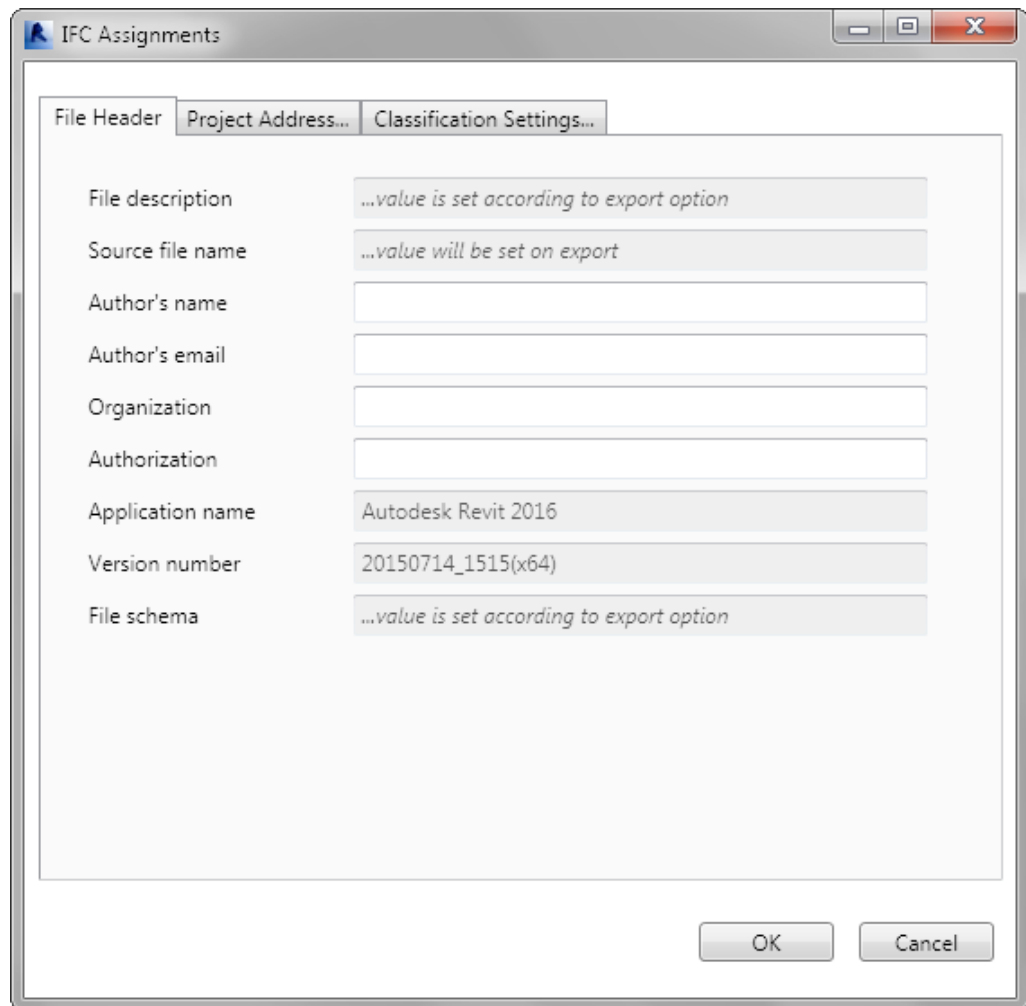


Fig. 3-46 - IFC export-assignments panel

### 3.3.3.3. IFC MODEL ANALYSIS

#### 3.3.3.3.1. GEOMETRY MONITORING WITH BIM-VISION

The result of this exportation can be checked using a IFC viewer. This step allows to check the effective and correct exportation of the geometric representation and the properties attached to the object.

On internet there is a diverse choice between different IFC viewers, some freeware, other with property formats.

In this case we chose BIMVision, a freeware IFC model viewer. It allows to view the virtual models coming from CAD systems like Revit, without necessity of having commercial licenses of these systems or having each of particular vendor's viewer. It has also many built-in features and is the first viewer with plugin interface. BIM Vision visualizes the BIM models created in IFC format 2x3 and his successive upgrades such as the "Coordination view 2.0" used among

this project.

In the main frame its there is the 3D viewer and in the right side we have the file browser and below the property viewer. In the upper part, different tabs allows to edit the object visualization or customize the user interface.

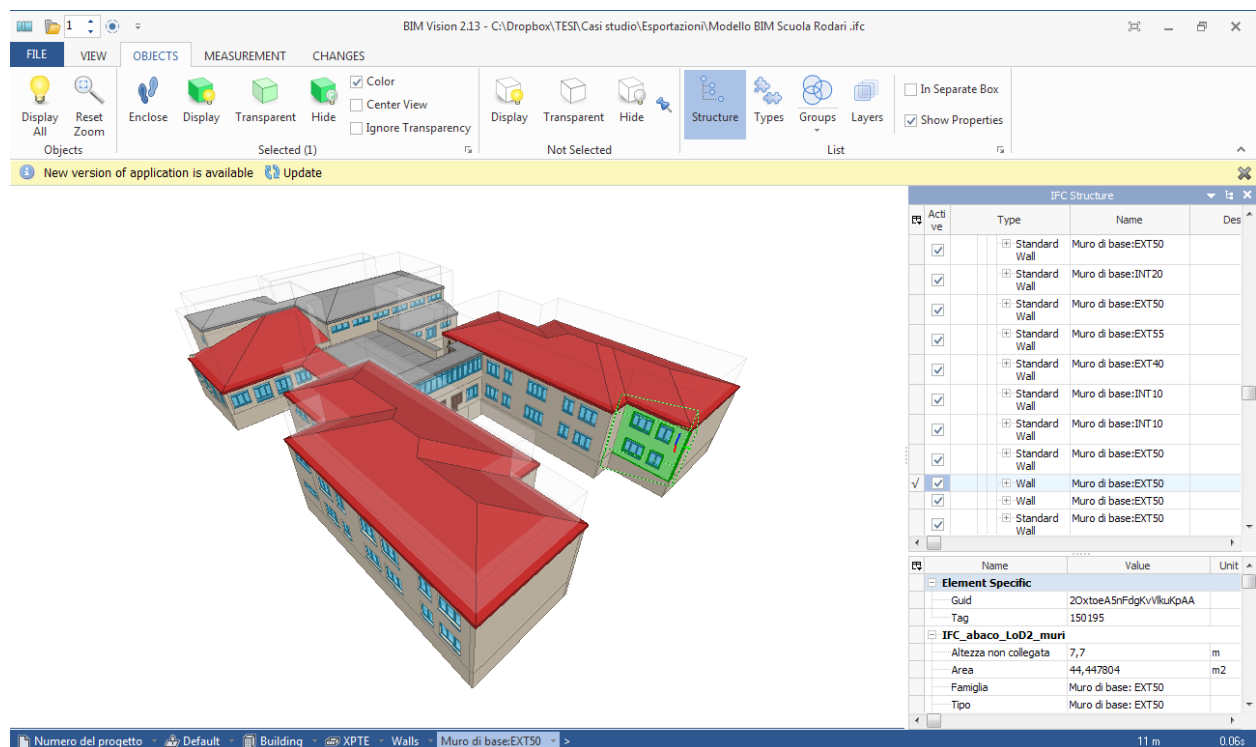


Fig. 3–47 - BIMVision interface

### 3.3.3.3.2. PROPERTIES MONITORING WITH IFC-QUICKBROWSER

Once checked the geometrical exportation of the IFC model we can start analysing it by a data point of view.

The IFC exchange file exported is a simple text document and so it is very easy to open and read it. But, due to its high complexity and the difficulty to understand the relations and the hierarchies between the different entities, we have to use a software called IfcQuickBrowser which allows to face these difficulties.

The software looks divided in two halves:

- Main window: the upper half displays the IFC model as it's exported. Double clicking on a string, if there any subtype attached to this line, these will appear on a tree disposition;
- Inverse window: the lower half displays the supertypes relative to the line selected in the main window. Double clicking on a string in the inverse window, if there any subtype attached to this line, these will appear on a tree disposition. Double right-clicking on a string, it will be highlighted the same entity but in the main window.

IfcQuickBrowser offers also a simple "Search" feature useful to control the effective exportation of a specific element or to navigate in the model filtering it with a specific keyword.

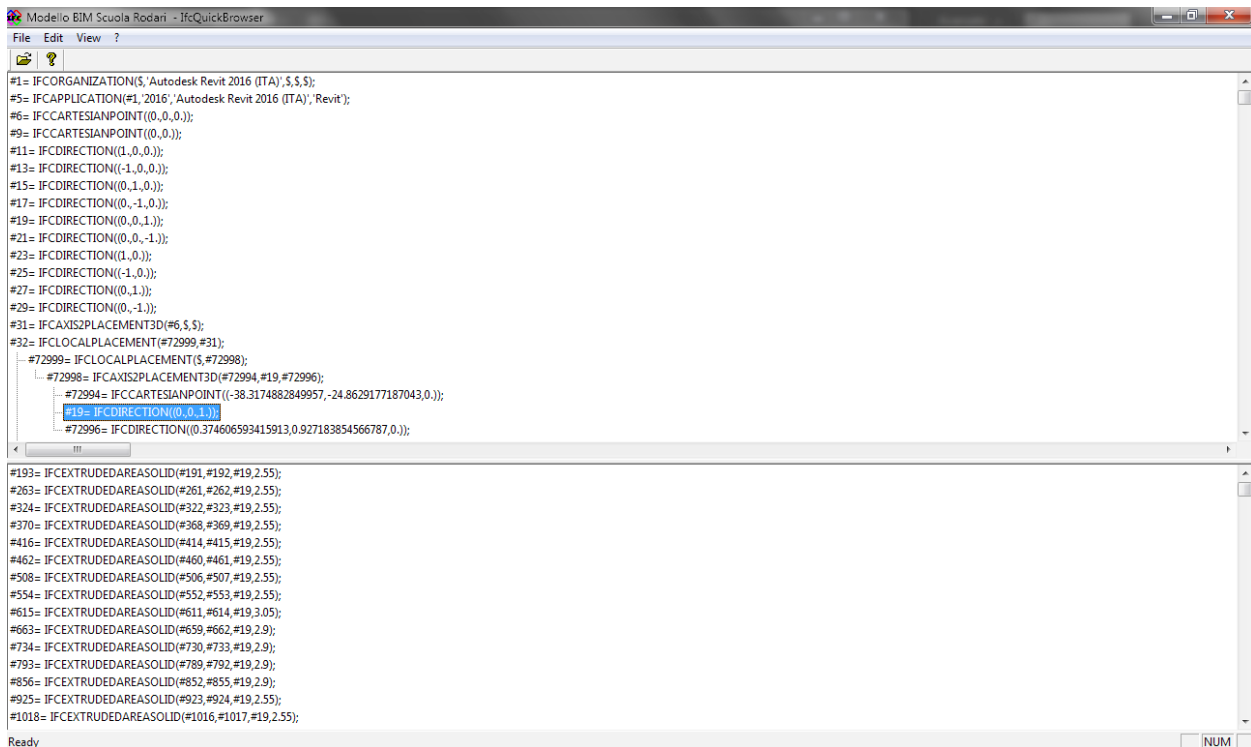


Fig. 3–48 - IfcQuickBrowser interface

The use of this specific software has proved to be fundamental to understand the great complexity and variety of elements exported from Revit to the IFC. The IFC standard used to exchange files in this project is the "IFC 2x3 Coordination view 2.0" which presents itself a remarkable reduction and simplification of the IFC entities number from 653 to 329, compared to the IFC 2x3 original standard. Beyond that, IFC is a very intricate exchange format and we must have well in mind his structure in order to not get lost between his code lines and to not waste time analysing useless informations.

To impose a filter over the multitude of IFC's informations of the building model, is also needed a general knowledge of the Revit model structure ad its nomenclature, in order to easily recognise in the IFC exchange format some specific entities and their relevance in the development of the standard definitions.

### 3.3.4. CONNECTIONS BETWEEN REVIT MODEL AND EXCHANGE MODELS

#### 3.3.4.1. ELEMENTS' IDENTIFIERS

A GUID is a 128 bit number, commonly split up into several fields of varying lengths and written in groups of 8-4-4-4-12 hexadecimal characters, i.e. 32 characters to represent the 16 bytes or 128 bits.

Revit's GUID, UniqueID and Element ID

On the other hand, we have the Revit UniqueID. Every element has such a unique identifier, which is returned through the model as a string. This string is formatted in groups of 8-4-4-4-12-8 hexadecimal characters. It is thus similar to the standard GUID format, but has 8 additional characters at the end. These 8 additional hexadecimal characters are large enough to store 4 bytes or a 32 bit number, which is exactly the size of a Revit Element ID.

In the gbXML tree structure, the CADObjectId element is used to map unique CAD object identifiers to gbXML elements. Since every Surface element own a CadObjectId sub-element, it allows external tools to read results from a gbXML file and map them to their Revit objects.

gbXML's CadObjectId element

This element contains a text string reporting the name of the Revit's type of object and, at the end, into square brackets, a sequence of 8 decimal characters representing the Revit element ID.

According to a detailed analysis of the IFC model made through IfcQuickBrowser, it is possible to individuate many connections between the objects contained in the Revit model and the IFC entities.

IfcGloballyUniqueId attribute and element ID in the IFC model

Once an IFC instance is created, it's fundamental that it can be uniquely identified for its entire lifecycle, in order to guarantee the ability to both exchange and archive data in the IFC model.

The IfcGloballyUniqueId is an IFC defined type that is a fixed length string value, created to contain this value as a compressed Globally Unique Identifier (GUID). The IFC2x specifications contains a public domain algorithm that can be used to create, compress and decompress these values.

For file-based data exchange, a methodology was devised to compress these GUIDs to conserve space when physically exchanging IFC models through various media. Given that each IFC object instance required a unique identifier containing a 128-bit number, a base 64 character encoding was devised as shown below:

0	1	2	3	4	5	6
0123456789012345678901234567890123456789012345678901234567890123						
0123456789ABCDEFGHIJKLMN	OPQRSTUVWXYZ	abcdefghijklmnopqrstuvwxyz	_	\$		

Fig. 3–49 - IFC-GUID Base-64 character encoding mapping

The resulting IFC-GUID is a fixed 22 character length string. Software implementations will need to use an algorithm that converts standard GUIDs to and from this encoding for compliance with the IFC specifications.

That said, the IFC GUID is also unique across sessions. This is one of the requirements of IFC. If I export a wall in one session via IFC, close the session, reopen the document, and reexport, the GUID remains the same for that wall, and should be different from any other wall's GUID.

Every time an IFC instance is created, this algorithm must be used to create a new `IfcGloballyUniqueId`. Re-use of an `IfcGloballyUniqueId`, even if the local instance is deleted, is strongly discouraged due to the inability to ensure that remote instances do not exist.

In the IFC model, it's also possible to have a reference about the Revit Element ID, which is contained two times in the attributes' set of every `IfcBuildingElement` entity in the model: once contained in the Revit's name string attribute after commas and suddenly as a specific standalone attribute.

### 3.3.4.2. ID REFERENCES' CHECKING

Direct checking through  
the exchange models

To begin this procedure we can start picking from the gbXML file a generic Surface element and extrapolate, from its `CadObjectId`, the 8 characters number contained into square brackets. In this case it has been selected a `InteriorFloor` type of surface and its `CadObjectId` declares the link with the "Pavimento:INT35" family and type of ceiling created on Revit.

```
<Surface surfaceType="InteriorFloor" constructionIdRef="aim7262" id="aim8841">
  <AdjacentSpaceId spaceIdRef="aim0854" />
  <AdjacentSpaceId spaceIdRef="aim3643" />
  <RectangularGeometry id="aim8842">
    (...)
  </RectangularGeometry>
  <PlanarGeometry>
    <PolyLoop>
      (...)
    </PolyLoop>
  </PlanarGeometry>
  <CADObjectId>Pavimento: INT35 [866472]</CADObjectId>
  <Name>T-9-52-I-F-73</Name>
</Surface>
```

The 8 characters number highlighted in red is the Revit Element ID described in the previous paragraph. Now we can check if this code is declared also in the IFC exchange model and to do this we can make a simple search operation, opening the IFC model as a text file with Ifc Quick Browser.

```
#59127= IFCSLAB('3FcsL$BIf1uvPP9o000$tS',#41,'Pavimento:INT35:866472',$. 'Pavimento:INT35',#59056,#59125,'866472',.FLOOR.);
```

The result is positive since it gives us the correspondence with a IfcSlab entity, belonging to the same Revit family declared in the gbXML's CadObjectId. Furthermore, the IfcSlab entity class declares firstly an attribute, highlighted in green above, which coincides with the IfcGuid described in the previous paragraph.

The next step is to verify the presence and the coherence of these ID in the BIM model using the Revit application.

Checking on Revit

We must first of all open the "IFC\_vista3D" view created as described in "3.3.3.2.1. Preliminary operations on Revit". Then, opening the Manage tab, we must click on the "Select Elements by ID" tool.

A window will open, where it'd be possible to paste in the empty field, the Element ID observed in the gbXML and IFC text files.

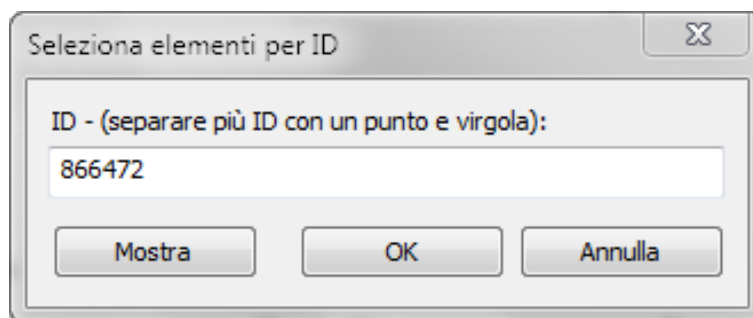


Fig. 3–50 - Revit's Select Elements by ID window

Automatically, the requested object is selected and, in order to highlight it from the multitude of other elements present in the view, we can use the "Isolate element" function selectable in the View Control Bar placed above in the main window. The result will be the following.

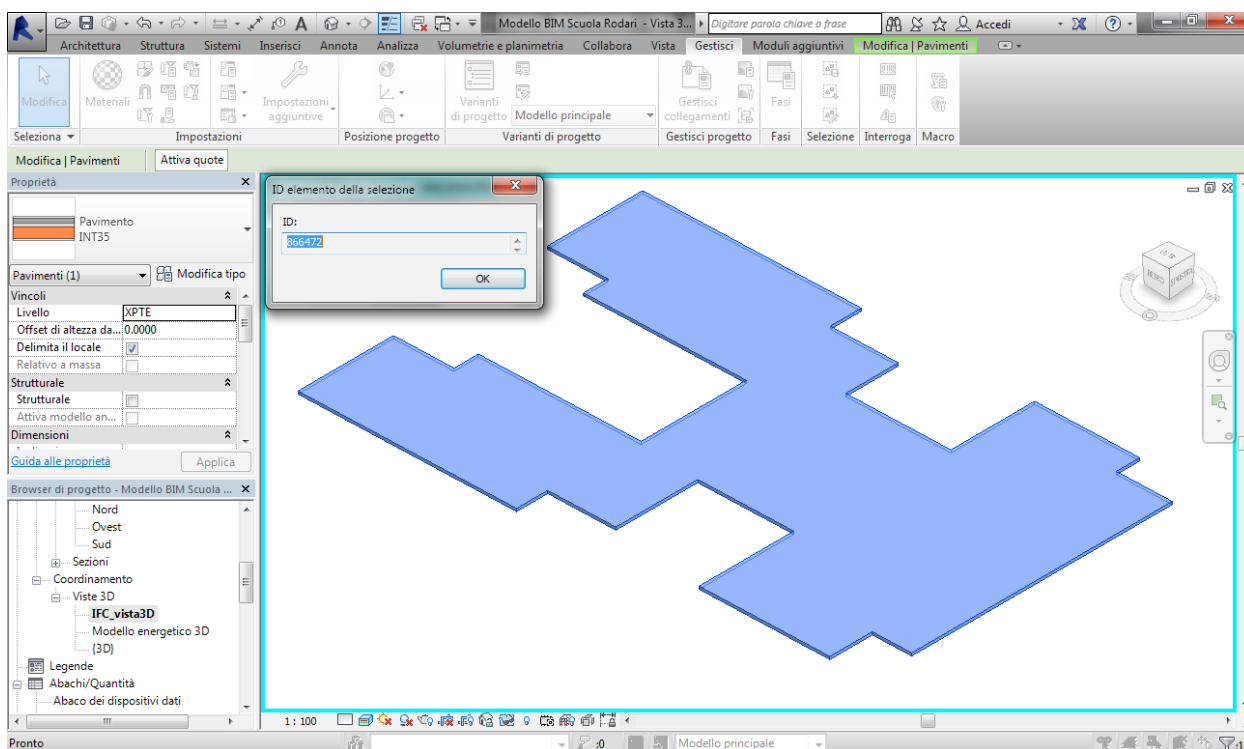


Fig. 3-51 - Revit's Element ID checking using the Select Elements by ID command

In this way, we can verify that the object previously selected and analysed in the gbXML and IFC models, is in fact a ceiling.

Looking at the Property panel of the selected object, we can check that it belongs to the "Floor" family and to the specific "INT35" type, as declared in the previous steps. In addition, we can also find in the panel the presence of a "IfcGUID" parameter, automatically created in the "IFC Parameter" once the exportation is done, which declares exactly the same 22 characters number, previously highlighted with green text, corresponding to the IFC GUID analysed before.

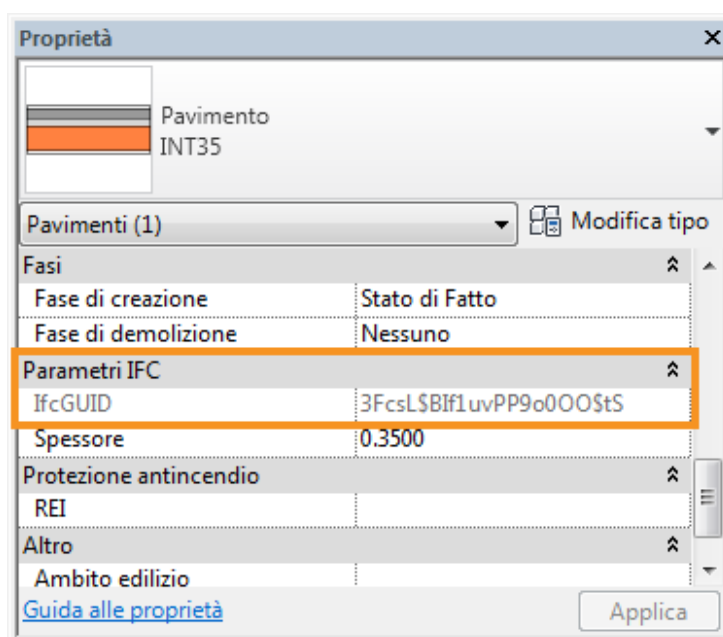


Fig. 3-52 - IfcGUID checking in the Revit element's property



Once we have analysed the IFC model as a text file, we can verify graphically, using BIMVision, if these ID references correspond and are correctly reported. Taking in mind the position in the model of the ceiling involved and using the tree map offered by the application, we can easily find and select the object in question.

Checking on BIMVision

In the property panel in the right-bottom corner, we can find under the "Element specific" grouping two properties, "Guid" and "Tag", which declares respectively the IfcGUID and the Element ID.

Once we have isolated the selected object using the View controls, the interface would appear like this.

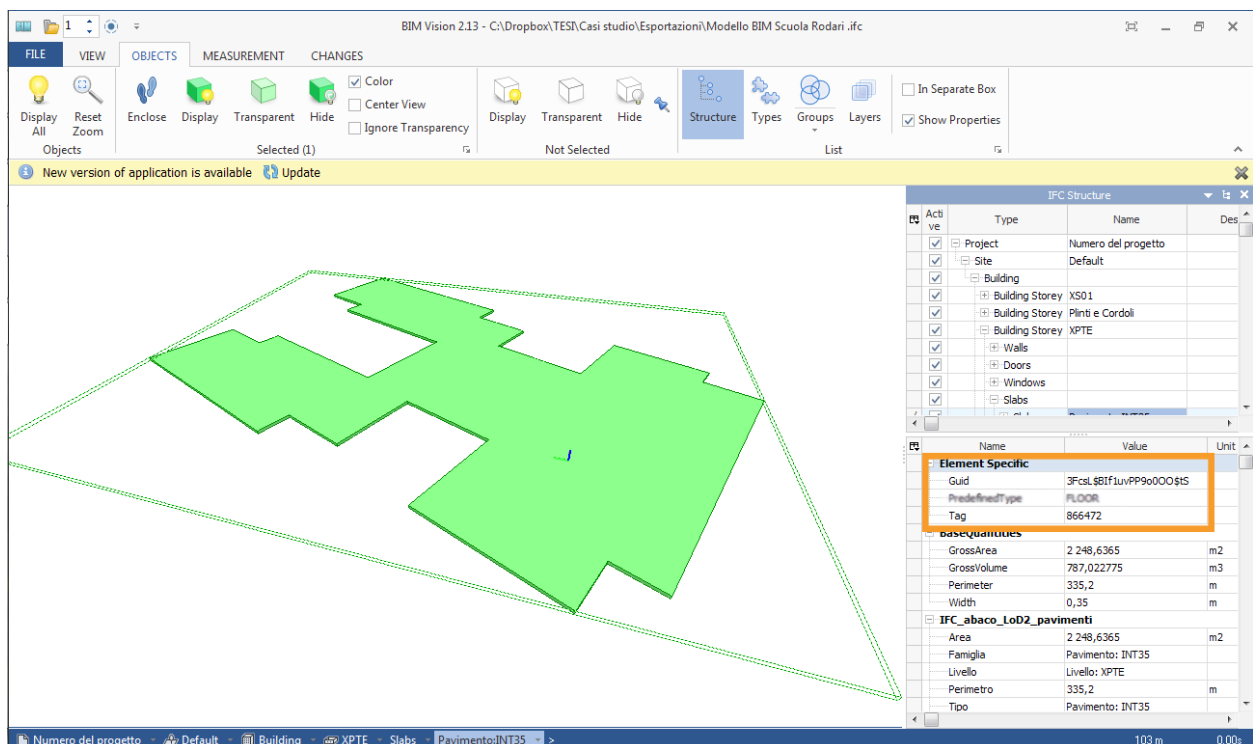


Fig. 3-53 - IfcGUID and Element ID checking on BIMVision

### 3.3.4.3. ADVANTAGES FOR THE INTEROPERABILITY

These ID encodings and then the connections between the entities in the different exchange models have been discovered and developed while mapping the gbXML and IFC formats, and it was immediately clear that this aspect could have a leading role in improving the communication and the interoperability during along the workflow.

The importance of these sequences of letters, numbers and symbols stands in the capability of representing, even if in various way, the same object with unique and univocal names. While the Revit objects' naming declares only the family or the type which the entity belongs, the previously analysed encoding refers

to specific objects inserted in the 3D model, enabling to track single building elements and data along the interoperability process.

In this way it's possible to combine the information in the various models, in order to attach them to the same single entity, giving it a limitless data richness in the hypothesis of sharing a infinite BIM compatible exchange models.

In the limited case of the present project only two exchange models have been used and analysed, but this gives us the possibility of combine them in order to find the more satisfying combination of geometries and data visualization.

The chart illustrated below represents a summary of the different ID numbers used to allow the communication between the models.

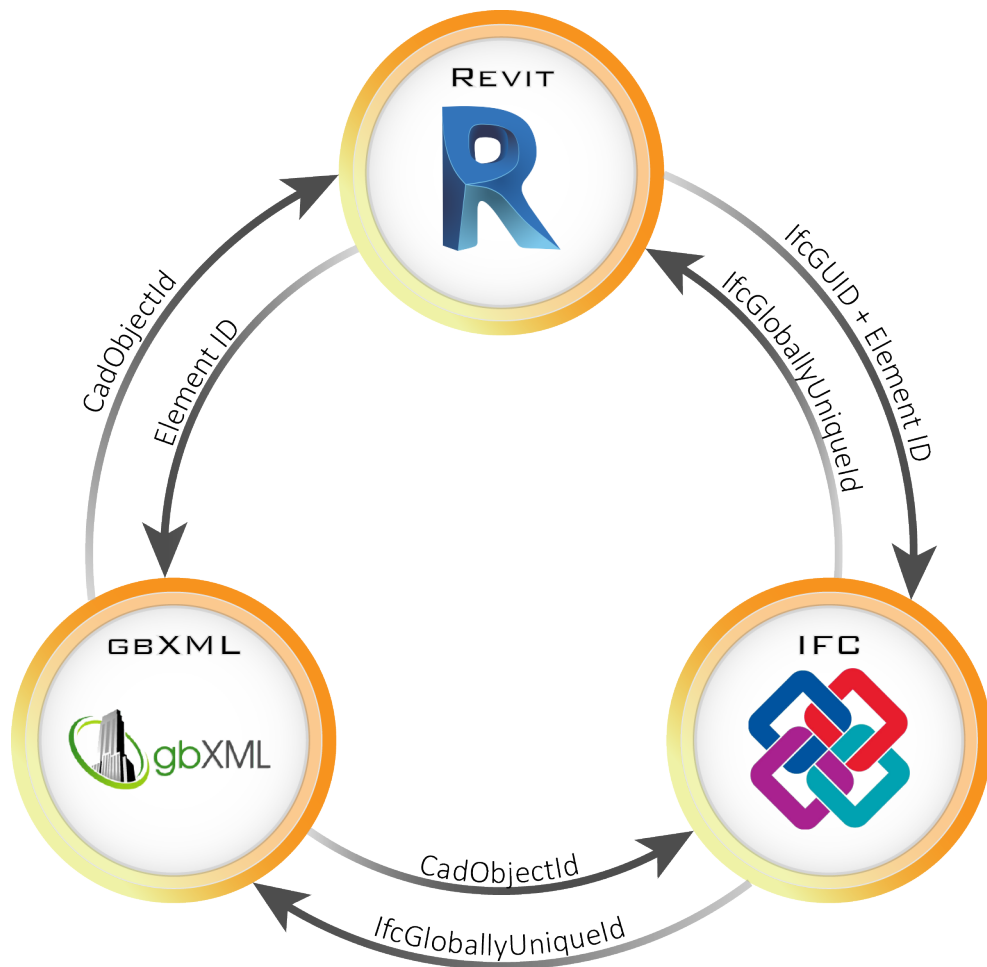


Fig. 3-54 - Overview of the communication between Revit model and exchange models

## 3.4. DIM PHASE

This final phase of this methodological workflow includes all the operations necessary to import the BIM models to the GIS platform and finally reach the proposed objectives.

The name of this phase refers to what introduced in "1.1. BIM/GIS interoperability", because it is in this phase that we will finally combine the BIM models with the GIS world, represented in this case by a virtual globe visualization, and verify the effective interoperability between the two systems.

The starting point of this stage, is to define which virtual globe use as GIS platform for this research. This decision, fallen upon the Cesium virtual globe, lead us in addition to specify a particular way of 3D rendering specification (3DTiles) and to adopt a specific 3D data exchange format (glTF) which are analysed more accurately in the next paragraphs.



Fig. 3–55 - DIM environment definition

### 3.4.1. VISUALIZATION ON CESIUM

#### 3.4.1.1. INTRODUCING CESIUM

Cesium is a JavaScript library for creating 3D globes and 2D maps in a web browser, based on WebGL.

Cesium was founded by AGI (Analytical Graphics, Inc.) in 2011 as a cross-platform virtual globe for dynamic-data visualization in the space and defense industries. Since then, Cesium has grown into a 3D globe serving industries from geospatial and oil and gas to agriculture, real estate, entertainment, and sports.

In 2017, Bentley Systems joined AGI, creating in this way the Cesium Consortium, an organization where these two enterprises work arm in arm in the development of the Cesium universe, accelerating and supporting its development and sustainability.

Thanks to its WebGL foundation, Cesium is cross-platform and cross-browser,



Fig. 3–56 - Cesium logo

since it can be used from different devices, OS and web browsers. Cesium is also open-source, so is free for commercial and non-commercial use, making it completely different from a 3D globe like Google Earth based on proprietary source codes. It differs from it also because it is not a complete application targeted at end users. It requires programming to use and has a lot of potential for customisation and user added content.

As specified in "1.2. Virtual globes", there are many virtual globes that could theoretically satisfy our requirements. First of all, referring to what imposed when we fixed our objectives, is the need to utilize a web based platform, in order to work in a hardware free environment, obtaining in this way great improvement in terms of visualization and sharing capabilities.

Secondly, we need to use a open source environment, in order to be application free and allowed to customize the GIS environment for our purposes.

According to [34] these are the exclusive reasons to choose Cesium among the other web based and open source contenders:

- Extensive collections of libraries: Cesium includes an extensive collections of libraries, related to import data from multiple sources, mathematical computations, creation of 3D geometries, camera and flight control. Overall, it makes easier to work with complex 3D city models.
- Improved performance: Cesium works directly over WebGL, providing hardware-accelerated graphics. Due to this reason, performance can be significantly improved while handling large complex 3D city models.
- Ease of use: The development on Cesium is comparatively easy due to well structured codebase and documentation. Each library is very well explained with examples and tutorials. Cesium also includes an interesting application called "Sandcastle" which provides live coding on the browser.
- Open source: Cesium is an open-source Javascript library. As it is free for commercial and non-commercial use, it is possible to add new features modify the existing features as per research's requirements.
- Active community: Cesium involves an active discussion forum, discussing and working on several issues and improvement suggestions. It also ensures a release every month, incorporating the issues and suggestions, making it an up to-date tool.
- 3D GIS capabilities: Although the primary intention to develop Cesium was the visualization of geo-spatial data, its wide collection of libraries enables to perform various GIS analyses on the globe. Additionally, with the help of WebGL, it is also possible to perform computationally intensive algorithms on the globe.

3.4.1.2.1. INTRODUCTION

3DTiles is an open specification for streaming massive heterogeneous 3D geospatial datasets.

3D Tiles define a spatial data structure and a set of tile formats designed for 3D and optimized for streaming and rendering. Tiles for 3D models use glTF, the WebGL runtime asset format developed by Khronos, which the Cesium team heavily contributes to. 3DTiles has been created for streaming massive geospatial datasets where a single glTF model would be prohibitive, as for the current case of representing a district with hundreds or thousands of building.



Fig. 3-57 - 3DTiles logo.png

The name of this specification refers to the concept of "tile", which represents its innovative core. 3DTiles has in fact radically changed the way 3D scenes were buffered and rendered in 3D GIS environment.

The concept of tile take its origins from the 2D visualization, since it is the way maps are decomposed in square regions that, according to the elevation of the view, are loaded with different levels of detail.

3DTiles is the innovative transposition of this concept in the 3D visualization, in the sense that it's no longer a simple plane square object, but has a 3D representation, which can include inside several objects and "child" tiles.

The concept of subdivision of a 3D scene in tiles of different levels of detail, as simplified in Fig. 3-58, is the real powerful engine of the 3DTiles specification, since it's not strictly connected only to a pre defined parameter, for example, of elevation but has inside a intelligence. In other words, we can customize the 3D scene's rendering dividing it in several tiles and child tiles, according to the concepts of relevance and density of objects.

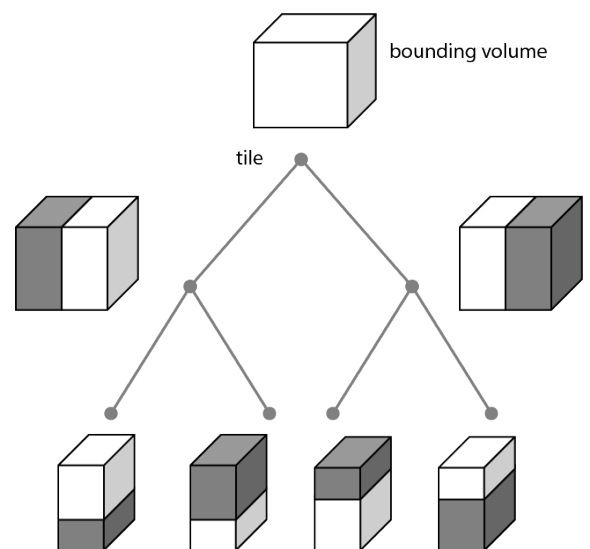


Fig. 3-58 - Tiles' tree structure

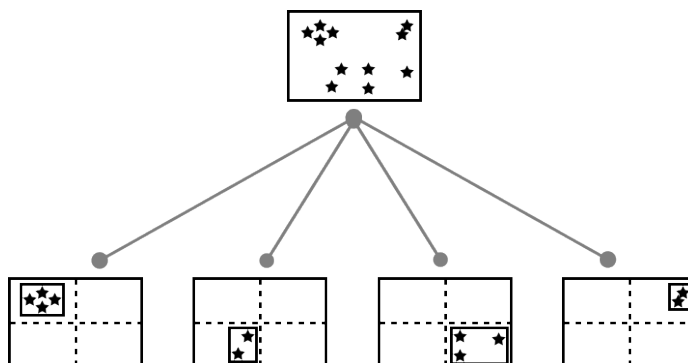


Fig. 3-59 - Subdivision of the scene in tiles and child tiles

In the image below it's possible to see in the Cesium 3D context the application of this concept: when a tile is contained in the view, then the set of all the elements contained inside it are streamed and loaded in the scene. When the tile can't be included in the window, the 3DTiles specification ignore it and suddenly looks for smaller tile, loading them as made before.

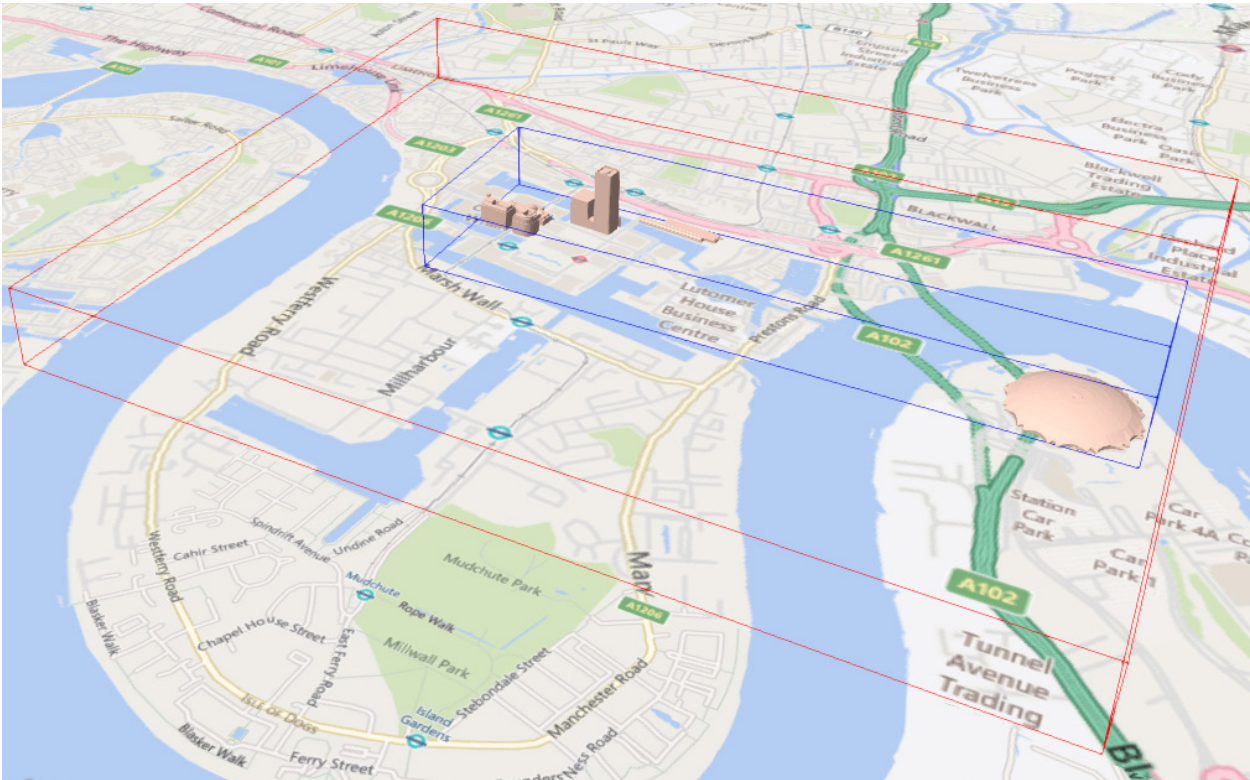


Fig. 3-60 - Tiles' representation on Cesium

This rendering mechanism can in this way bring many benefits in terms of memory usage and loading time spreading.

Besides this important innovation, 3DTiles bring other attractive features which allows a wide range of customization for interaction, such as highlighting on mouse over, or removing a 3D building. Tiles can contain metadata for each model to allow additional interaction, such as querying third-party web services using a building ID.

#### 3.4.1.2.2. SYNTAX AND STRUCTURE

##### SYNTAX

##### JSON inheritance

JSON, or "JavaScript Object Notation", is a lightweight text-based open standard designed for human-readable data interchange.

JSON's data organization is based on two data structures, which are fundamental to support communicability and interoperability between the different programming languages supported by JSON (C, C++, Javascript...):

- A collection of name/value pairs;
- An ordered list of values.

The first kind of structure, in JSON is represented by an entity called "object". An object is an unordered serie of names / values; it starts with "{" and ends with "}" symbols. Each name is followed by ":" preceding the "value" attribute and different name-value pairs are separated by a comma.

A value can be of different kind: string in double quotes, or a number, or true or false or null, or an object or an array. A string is a sequence of Unicode characters wrapped in double quotes.

The second kind of structure is represented in JSON by "array" entities which consists in a sequence of values separated by commas.

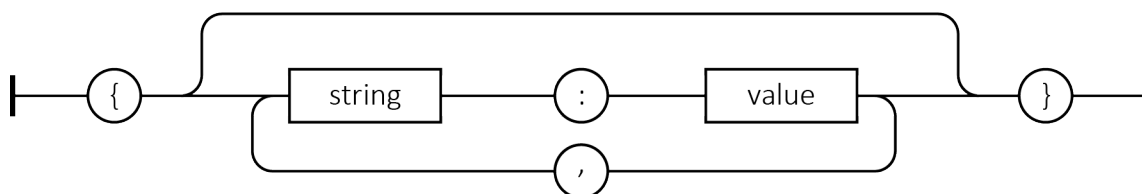
Here follows an example of a sequence of data storage in JSON's syntax.

```
{
  "book": [

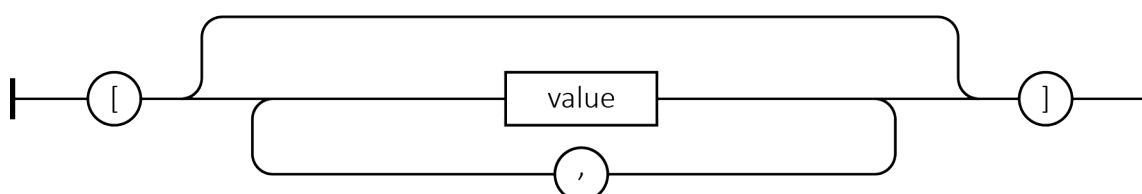
    {
      "id": "01",
      "language": "Java",
      "edition": "third",
      "author": "Herbert Schildt",
    },

    {
      "id": "07",
      "language": "C++",
      "edition": "second",
      "author": "E.Balagurusamy",
    }
  ]
}
```

#### OBJECT



#### ARRAY



VALUE

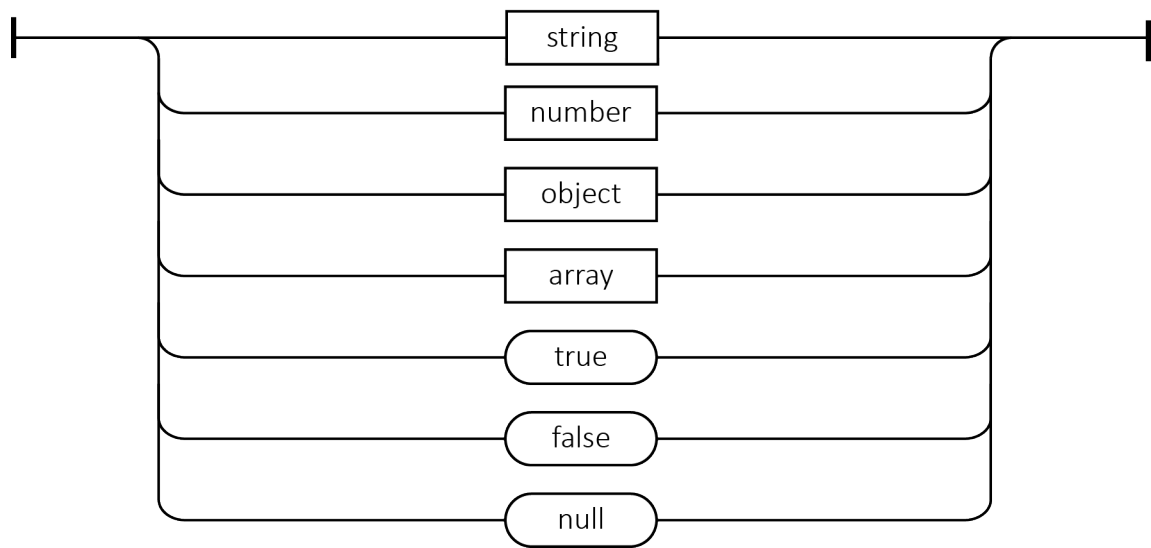


Fig. 3–61 - JSON's main entities definition

### STRUCTURE

#### tileset.json

The structure of a 3DTiles scene's streaming is regulated first of all by a JSON based text file called tileset.json.

This file is composed by a set of sub-objects which defines the tiles' visualization and content. In particular, the "root" object specifies the characteristics of the upper tile, describing with the "region" attribute its 3D perimeter and with the "content" attribute the list of child objects contained inside. These objects can be sub-tile objects (under the "child" attribute) or be referred, through the "url" attribute to specific 3D models to be loaded when the current tile has to be rendered.

#### Batched 3D model

These models are imported in the tileset.json file with the .b3bd extension, that describes a particular 3D data asset for gathering in the same batch geometrical informations and specific model metadata.

Analysing its byte-structured layout, it's possible to divide it in two sections:

- a "Header" part, where the "batchTableJSONByteLength" and the "batchTableBinaryByteLength" sections contains, respectively in JSON and binary format, the metadata associated to the geometric entities;
- a "Body" part, where the "Binary glTF" section gathers the 3D geometric model.

This asset is very useful for representing in 3DTiles BIM based exchange models as those we have exported from Revit in gbXML and IFC formats, so that we



can maintain the duality geometry-attribute, very important in terms of final assesment of the interoperability process.

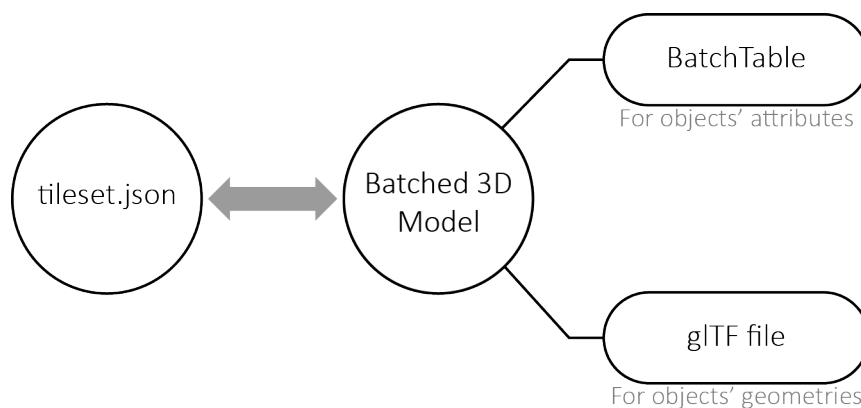


Fig. 3–62 - 3DTiles' files organization

### 3.4.1.3. GLTF FORMAT

#### 3.4.1.3.1. INTRODUCTION

glTF, which stands for "GL Transmission Format", is a 3D model exchange format developed by Kronos Group.

Kronos Group is a non-profit consortium formed in 2000 that deals, among other things, with the development of open 3D standards and specifications.

Its structure is based upon the COLLADA transmission format, which is considered as one of the most advanced 3D asset interchange format, and it is also distributed by Kronos Group.

COLLADA is a XML database schema for 3D assets and it can hold everything to do with a virtual scene: geometries, animations, advanced materials and visual effects, physical properties, etc.

glTF was born from the necessity of creating a asset transmission format for rich native and Web 3D applications, which could connect the world of 3D data exchange formats (which belongs to COLLADA) and that of 3D execution platforms, in particular all the GL APIs (represented in this research by Cesium, which is WebGL based).

glTF is considered as the "JPEG of the 3D" [43] for its great versatility and compressibility, making it a more and more diffused format among GL API. In short, its main advantages are the followings:

- Compact to transmit;
- Fast to load;
- Describes full scenes;



Fig. 3–63 - glTF logo

Bridge upon 3D asset interchange formats and GL APIs

glTF's advantages

- Runtime neutral;
- Open and extensible.

Beyond these benefits, the choice of this exchange format is clearly subdued to the decision made upstream (as described in "3.4.1.1. Introducing Cesium") to choose Cesium as virtual globe for this project.

glTF is in fact, together with 3DTiles, the native supported format in Cesium, and this means that the implementation is constantly supported by the Cesium Consortium, in order to guarantee its performances in their virtual globe.

Other alternatives have been considered, including KML, which has been for years the leading file format in the virtual globes market following the development of Google Earth and Google Maps.

KML is a XML based language, and this makes it a prolix standard compared to glTF: KML is in fact a high level of abstraction language, in which data are packed in a sophisticated and heavy structure, while glTF is based, apart from the JSON header file (see next paragraph) on a binary format meant to mirror the GPU APIs as closely as possible, and this feature allows not to require conformance requirements for an implementation to stream data versus downloading it in its entirety before rendering. glTF is this way immediately "ready" for use, obtaining great advantages in terms of implementation and of time and memory employment, while KML, who was developed for Google exclusive application, has difficulties in terms of global implementation and WebGL application.

Tab. 3-3 - KML and glTF comparison

KML	GLTF
XML based	Binary
High abstraction	GPU APIs alike
Heavy format	Light format
WebGL low compatible	WebGL high compatible

KML has become in the last period a supported format by Cesium, but its implementation isn't at the same advanced level as for glTF [33].

### 3.4.1.3.2. SYNTAX AND STRUCTURE

#### SYNTAX

A glTF asset is a combination of different files; specifically, it is represented by:

- **A JSON-formatted file (.gltf)** containing a full scene description: node

hierarchy, materials, cameras, as well as descriptor information for meshes, animations, and other constructs;

- **Binary files (.bin)** containing geometry and animation data, and other buffer-based data;
- **Image files (.jpg, .png, etc.)** for textures.

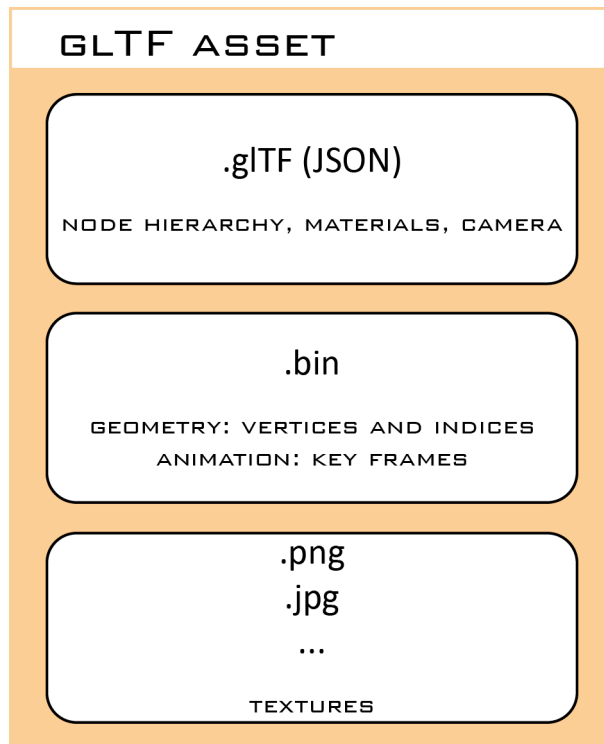


Fig. 3–64 - glTF asset's composition

The core of glTF is the JSON file that describes the structure and the composition of a scene containing 3D models.

## STRUCTURE

Focusing only on the glTF asset, we can make a description of how the informations and the geometries are organized inside it.

Above all, every glTF asset must contain one and only "asset" object. This element allows to find information about the glTF version, which specifies the target glTF version of the asset.

"asset" object

```
{
  "asset": {
    "version": "2.0",
    "generator": "collada2gltf@f356b99aef8868f74877c7ca545f2cd206b9d3b7",
    "copyright": "2017 (c) Khronos Group"
  }
}
```

After this element is declared, the .gltf file is organized in a tree structure in

which the top-level element are listed and connected each other as it follows.

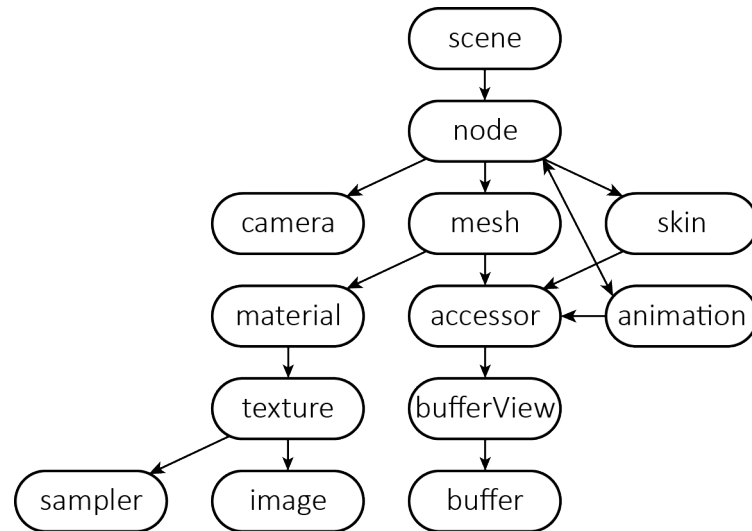


Fig. 3-65 - glTF asset's structure

#### Scene and nodes: basic structure of the scene

The "scene" object declares the set of visual objects to render, organized in a "array" structure.

Each scene may contain one or more "nodes" object, which represent the single object contained in the correspondent "scenes" entity. Object are organized in a parent-child hierarchy known informally as the node hierarchy, in which the sub-level object "children" links the child nodes to the parent node.

```

{
  "nodes": [
    {
      "name": "Car",
      "children": [1, 2, 3, 4]
    },
    {
      "name": "wheel_1"
    },
    {
      "name": "wheel_2"
    },
    {
      "name": "wheel_3"
    },
    {
      "name": "wheel_4"
    }
  ]
  "scenes": [
    {
      "name": "singleScene",
      "nodes": [
        0
      ]
    }
  ],
  "scene": 0
}
  
```

Any node can contain one "mesh" object, defined in its mesh property. Mesh can be skinned using a information provided in referenced "skin" object.

In glTF, meshes are defined as arrays of primitives, which specify one or more "attributes" object, corresponding to the vertex attributes. Indexed primitives also define an "indices" property. Attributes and indices are defined as references to accessors containing corresponding data. The material that should be used for rendering is also given, by the index of the material.

Each attribute is defined by mapping the attribute name to the index of the "accessor" object that contains the attribute data. This data will be used as the vertex attributes when rendering the mesh. The attributes may, for example, define the POSITION and the NORMAL of the vertices.

```

"meshes": [
  {
    "primitives": [
      {
        "mode": 4,
        "indices": 0,
        "attributes": {
          "POSITION": 1,
          "NORMAL": 2
        },
        "material": 2
      }
    ]
  }
],

```

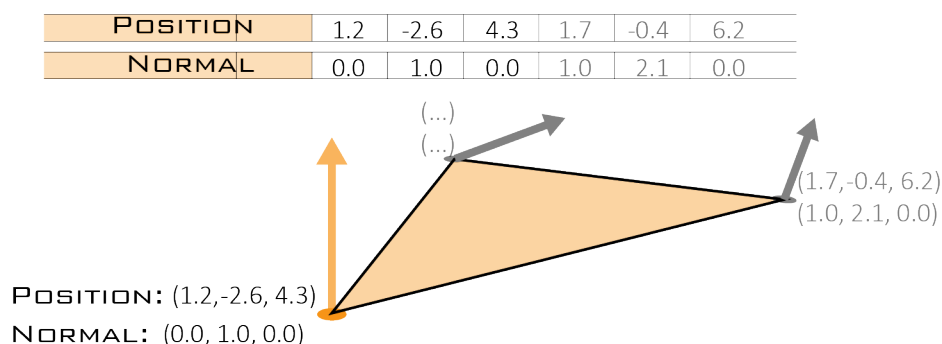


Fig. 3-66 - glTF geometries' representation

## 3.4.2. CONVERSION TO GLTF FORMAT

### 3.4.2.1. CONVERTER SOFTWARE USE

Once the gbXML and the IFC exchange models are loaded in the Middleware, they must be converted in order to be exported on Cesium.

This is made through the use of a software developed in parallel to this thesis according to the indication received by the analysis made on the exportation format used.

This software is composed of three parts:

- IFC parser: converts the IFC model in a 3D mesh surfaces with associate attributes;
- gbXML parses: converts the gbXML model in 3D mesh surfaces with associate attributes;
- 3DTiles generator: creates a tileset.json file and a B3DM asset for each model we want to render in the tile.

The Ifc Parser, gbXML parser and 3D Tile Generator are implemented with Oracle Java Platform, Enterprise Edition 8 (Java EE 8) technology [46]. The names are actually in alpha status and are part of a code base named actually "webglobes.org".

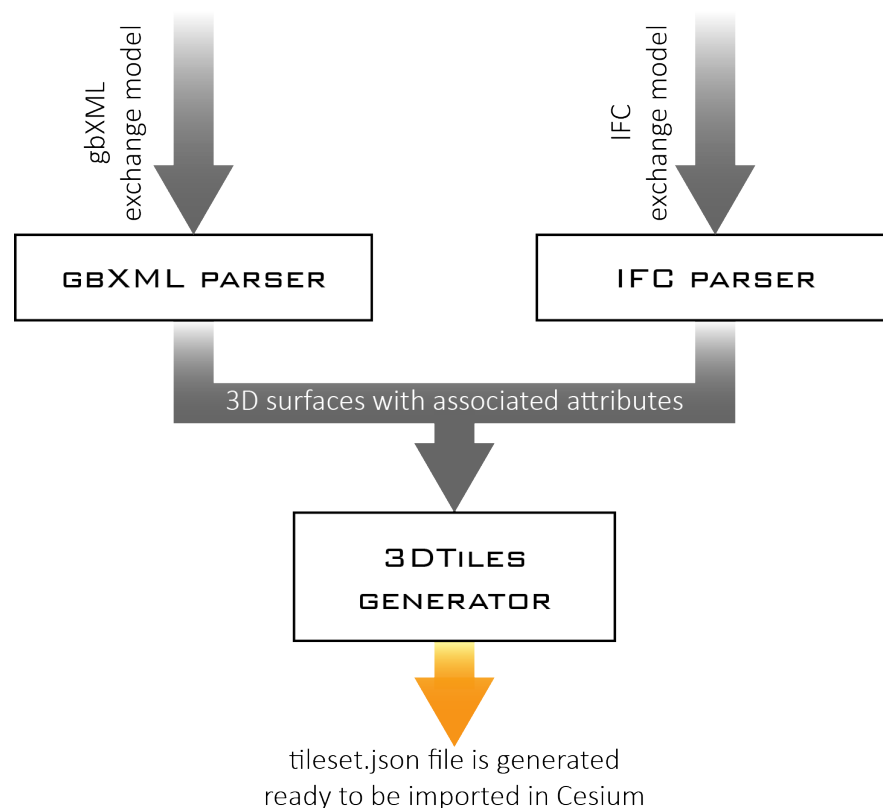


Fig. 3-67 - Overall conversion process schema

#### 3.4.2.1.1. IFC PARSER ALGORITHM

The Ifc Parser is a set of methods that parses an Ifc model and produces a generic 3D representation:

- 1) The IFC file is parsed with the help of the JSDAI library in a set of object oriented Java classes. The Java classes represent as a tree of properties the original structure of data of the original file
- 2) The tree of properties is queried to extract the subset of data needed:
  - General properties of the model, for example reference system, coordinate type, geodetic coordinates;
  - The entity of type IfcSite, and the tree in the form of deeper lists of IfcBuilding, IfcStorey and IfcProduct (see Fig. 3–33).

Actually the supported classes of IfcProduct are IfcWall, IfcWallStandardCase, IfcOpeningElement, IfcDoor, IfcWindow, IfcRoof, IfcSlab, IfcBuildingElementProxy and IfcFlowTerminal

- 3) For every IfcProduct supported is verified if is present in the model a suitable representation, mainly of type IfcShapeRepresentation. If the representation found is supported, the 3D geometry of the product is built according to the definition of the geometric model standard in CSG form (by means of the JCSG library) or as a list of surfaces with or without holes, according to the original geometry;

The supported representations of the product are IfcMappedItem, IfcFacetedBrep, IfcClosedShell, IfcFace, IfcFaceouterBound, IfcFaceBound, IfcLoop, IfcPolyLoop, IfcCartesianPoint, IfcExtrudedAreaSolid, IfcBoundingBox, IfcPolyline, IfcBooleanClippingResult, IfcPolygonalBoundedHalfSpace, IfcHalfSpaceSolid, IfcGeometricSet, IfcPlane, IfcCurve, IfcTrimmedCurve, IfcBoundedCurve, IfcCompositeCurve, IfcTrimmedCurve, IfcCircle, IfcTrimmingSelect, IfcProfileDef, IfcRectangleProfileDef, IfcArbitraryClosedProfileDef and IfcCircleProfileDef.

To every product is associated the list of IfcMaterial(s) as a set of colours, transparencies and material definitions as sizes and construction definitions, and related alphanumeric attributes.

- 4) Every surface is then subjected to a list of ordered coordinate transformation according to the IFC spec;

At the end of this step the model geometry is fully reconstructed in the original coordinate system and every element has a set of alphanumeric attributes associated.

The work of the Ifc Parser is done and the control flow goes to the 3D Tiles Generator

#### 3.4.2.1.2. GBXML PARSER ALGORITHM

The GbXML Parser is a set of methods that parses a model in GbXML format and produces a generic 3D representation:

- 1) The GbXML file is parsed with the help of the JAXB library in a set of object oriented Java classes. The Java classes represent as a tree of properties the original structure of data of the original file;
- 2) The tree of properties is queried to extract the subset of data needed:
  - General properties of the model, for example reference system, coordinate type, geodetic coordinates;
  - The entities of type Wall (external and/or internal), Roof and Opening.
- 3) For every entity supported the correspondent 3D geometry is built according to the definition of the geometric model as a list of surfaces with or without holes. To every product is associated the list of Material(s) as a set of colours, transparencies and material definitions as sizes and construction definitions, and related alphanumeric attributes (very limited in number and definition with respect to the IFC case).

At the end of this step the model geometry is fully reconstructed in the original coordinate system and every element has a set of alphanumeric attributes associated.

The work of the GbXML Parser is done and the control flow goes to the 3D Tiles Generator.

#### 3.4.2.1.3. 3D TILE GENERATOR ALGORITHM

This module has the aim to build a structure of text (tileset.json index file) and binary (Batched 3D Model files) according to the 3D Tiles specification:

- 1) The Tile Generator receives in input a list of 3D surfaces associated with attributes and materials. The first step is to clean the surfaces from eventually duplicated points and to perform a rotation of the model on the z axis;
- 2) The surfaces are organized and indexed by material for optimized visualization purposes;
- 3) The 2D surfaces (eventually with holes) are triangulated by the Poly2Tri library to obtain a list of triangles suitable for the visualization, every triangle is indexed and the vertex normal are computed;
- 4) The triangles are transformed from the local coordinate system of the model to the final geodetic position on the globe in terms of latitude and



longitude (and eventually rotation) according to the Cesium RTC extension;

5) The attributes of every surface are packed in a Batch Table suitable to be stored in the Batched 3D Model;

6) Indexes, positions and normals are stored in a GLTF structure and the final B3Dm model is built;

The steps from 2) to 6) are repeated according to the LoDs logic and definition requested

7) The `tileset.json` index file is generated according to the number of models produced.

### 3.4.3. IMPORTATION AND VISUALIZATION

#### 3.4.3.1. IMPORTATION OF THE MODEL IN CESIUM

After the exportation is done, in a directory folder we will find the `tileset.json` and the `b3dm` files generated by the software.

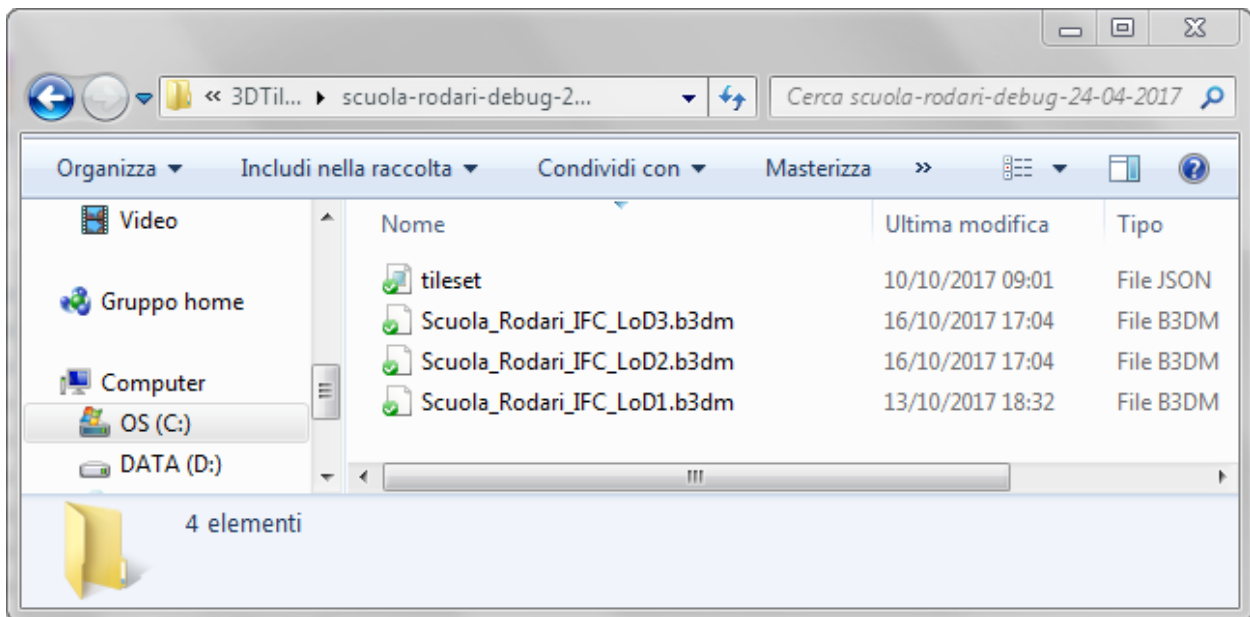


Fig. 3-68 - Folder containing the `tileset.json` and the `b3dm` files

Opening the `tileset.json` file it's possible to see how the 3DTile generator has set the rules of visualization of the 3DTiles scene, defining progressive child and children tiles which, through the "url" objects, call a specific `b3dm` file to be loaded when the tile in question is rendered

```
{
  "asset": {
    "version": "0.0"
  },
  "geometricError": 40,
  "root": {
    "boundingVolume": {
      "region": [
        0.13556779355972096,
        0.7877132234975514,
        0.13558027911822904,
        0.7877255569982139,
        -3.1498439812512053,
        3.8503479407402343
      ]
    },
    "geometricError": 16,
    "refine": "replace",
    "content": {
      "url": "Scuola_Rodari_IFC_LoD1.b3dm"
    }
  }
}
```

```

},
"children": [
  {
    "boundingVolume": {
      "region": [
        0.13556779355972096,
        0.7877132234975514,
        0.13558027911822904,
        0.7877255569982139,
        -3.1498439812512053,
        3.8503479407402343
      ]
    },
    "geometricError": 8,
    "content": {
      "url": "Scuola_Rodari_IFC_LoD2.b3dm"
    },
    "children": [
      {
        "geometricError": 4,
        "content": {
          "url": "Scuola_Rodari_IFC_LoD3.b3dm"
        },
        "boundingVolume": {
          "region": [
            0.13556779355972096,
            0.7877132234975514,
            0.13558027911822904,
            0.7877255569982139,
            -3.1498439812512053,
            3.8503479407402343
          ]
        }
      }
    ]
  }
]
}
}

```

This 3DTiles asset can now be loaded on the Cesium application, thanks to a HTML calls which load all the files to a specific web server.

### 3.4.3.2. OVERALL VISUALIZATION

When opening the web link, we are directed to a new browser window, where after a short time loading the scene is rendered.

The interface appears as it follows.



Fig. 3–69 - Dashboard interface

In the top-right corner it's possible to see the view's navigation tools.

First on the left we have a "Search" button, where we can type an address or a set of coordinates to go to a specific corner in the globe. Then, on its right, a "Home" button that, when clicked, shows directly a scene from a predefined camera. Suddenly, with the next button we can skip from different scene modes: 3D, 2D or Columbus view (which can be described as a "2,5D" perspective). With the following button we can instead choose between several maps layers and terrains. The last "Help" ("?") button opens a tab which gives instructions about the mouse and touch screen controls for navigate in the scene.

Pointing the mouse over an object, this will be highlighted and after a few seconds it will appear a cloud panel showing the properties attached to the object.

In bottom part of the main window it's possible to use a particular toolbar which allows to skip from a "real time" default visualization, changing hour or day of visualization. This is useful especially in order to change the shading appearance over the model.

#### Surrounding buildings visualization

But the most important aspect, except the loaded BIM model, is the visualization of the surrounding buildings which are created following the same procedure made for import the BIM model in Cesium as Batched 3d Models.

This was made firstly by recovering, from the Piedmont Region's website, the Municipality Technical Map as a shapefile (.shp). This format, which is the standard one for GIS software, is composed by a set of georeferenced geometries, especially polylines, and attributes attached to these.

This file has been converted to a B3DM asset after a shapefile parsing procedure, similar to that used for IFC and gbXML models: the buildings volumes have been created using the shapefile's polylines as base for extrusions, which heights have been referred to the "Eight" attribute attached to them. Other attributes, that are visible pointing the mouse to single buildings, have been gathered in Batch Table files.

We can finally notice the capability of the dashboard to be visualized from a smartphone, using a mobile web browser.

Smartphone compatibility



Fig. 3-70 - Dashboard visualization from smartphone



## 4. RESULTS ANALYSIS

## 4.1. VISUALIZATION

### 4.1.1. OBJECTS AND ATTRIBUTES REPRESENTATION FROM GBXML AND IFC MODELS

#### 4.1.1.1. FROM GBXML MODEL

##### LoD VISUALIZATION



Fig. 4-1 - LoD visualization of the gbXML based model

- **LoD1 visualization:** The object represented doesn't derive from the gbXML model but it's a simple extrusion derived from the district shapefile. This is due to the inability of gbXML to include a simplified representation of the model;
- **LoD2 visualization:** Only the ExteriorWall, Opening and Roof type of gbXML surface are represented;
- **LoD3 visualization:** the Roof surfaces are excluded, so that we can visualize the building in the inside. We can't include more objects for a more detailed rendering.



##### POSITIONING ON THE VIRTUAL GLOBE

The positioning of the model had positive results since the model is automatically placed on the imprint of the building showed by the satellite view layer.

The dimensions and the orientation also corresponds to the real building.



## VISUALIZATION ISSUES



Fig. 4-2 - gbXML visualization issues

As shown in the images above, this visualization presents some issues:

- **Transparent surfaces:** As we can see in the left picture, some exterior walls seem to be not represented. In reality they are included in the scene but with certain perspective appears like transparent surfaces. If we turn the camera, the wall in fact appears as a normal opaque surface. This issue is probably due to the manageability of the NORMAL attribute object of the glTF model of the building;
- **Discontinuity between surfaces:** As they are not properly building elements, the analytical surfaces are not required to have physical connections between them. This imply that in the 3D visualization these surfaces can appear disconnected.

### 4.1.1.2. FROM IFC MODEL

## LOD VISUALIZATION



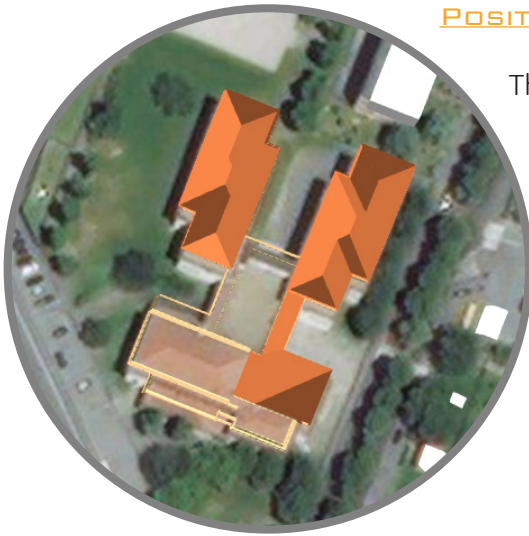
Fig. 4-3 - LoD visualization of the IFC based model

- **LoD1 visualization:** The object represented is exported from the IFC

model as a Revit mass, which corresponds to the IFC's IfcBuildingElementProxy class. This class allows us to use a single but realistic volume for representing the whole building with low detail;

- **LoD2 visualization:** The Building Elements represented belong to the IfcWall and IfcRoof classes; only the external walls are rendered, according to filter algorithm based on the IsExternal attribute, exported from the Revit model with the PSet property set.
- **LoD3 visualization:** the IfcRoof instances are excluded, so that we can visualize the building in the inside. Also the interior walls are now rendered.

#### POSITIONING ON THE VIRTUAL GLOBE



The positioning of the model had positive results since the model is automatically placed on the imprint of the building showed by the satellite view layer.

The dimensions and the orientation also corresponds to the real building.

#### PROPERTIES AND COLOURS ATTACHMENT

As shown in the side picture, the dashboard allows to get informations about the object highlighted pointing the mouse over it and visualizing then a cloud panel. These informations correspond to the Revit parameters defined LoD per LoD previously.

Also the colours used for the rendering derive from the Revit model, stored in the IFC model in the IfcColor class as RGB coordinates.





Fig. 4-4 - IFC visualization issues

- **Transparent surfaces:** As for the gbXML model, also in this case many geometries are represented as transparent surfaces in certain points of view. The problem is here accentuated by the fact that the objects derived from the IFC model are more complex and are composed of lots of planar surfaces.
- **Difficulty in representing complex objects:** Taking as a reference the example of the windows, the dashboard is currently unable to represent this kind of object, since it's composed by different element (glass, infix, etc.), each with a particular representation and its own reference system.
- **Geometric errors:** As we can see in the right picture, the represented model shows many errors in the generation of certain objects, due to a current incorrect reading of its Shape Representation.

## 4.1.2. COMPARISON AND BEST CASE CHOICE

### 4.1.2.1. IFC AND GBXML'S STRENGTHS AND WEAKNESSES

Once we analysed case by case basis the two final outputs, we can discern about the issues faced and highlighted above and make a comparison between them in order to find the most performing options.

But before analysing specific strengths and weaknesses of each option, it can be useful to make a introductive generic comparison between the IFC and gbXML formats, which can implicitly explain and justify the showed results.

IFC vs gbXML practical and theoretical comparison

Firstly, IFC adopts a comprehensive and generic approach to represent an entire building project, covering domains from building construction to building operation. IFC representation was also extended in the building commission domain and implemented in several cases studies. On the other hand, the application of gbXML, officially deployed by Green Building Studio Inc., is currently only on the energy simulation domain and this limits obviously its interchange's capabilities.

Secondly, IFC uses a "top-down" and relational approach, which yields in a relative complex data representation schema and a large data file size. gbXML adopts a "bottom-up" approach, which is flexible, open source, and a relatively straightforward data schema. The "top-down" approach can trace back all the semantic changes when one value of the element in the schema changed. Ideally, it has the ability to maintain semantic integrity automatically. However, it is very complex to program and be implemented in software. The "bottom-up" approach has less layer of complexity.

In other words, with a comprehensive "top-down" data schema, IFC shows potential benefits in its highly organized and relational data representation. In contrast, the "bottom-up" gbXML schema is simpler and easier to understand, and this facilitates quicker implementation of schema extension for different design purposes [36].

Planar surfaces vs detailed volumes

Once said this generalities about the two formats, the first assessment to make is the consideration of the difference between the geometries exported. While gbXML only exports rectangular shapes, which is enough for energy simulation, the IFC may export different kinds of representation. As analysed accurately in "3.3.3.1.2. Syntax and structure", also the same object in the IFC file may include different representation instances, both 2D and 3D. In shorts, on one hand we have only simple rectangular plane surfaces, on the other hand we can manage

less or more accurate tridimensional models: from simple extrusion to detailed bounding volumes.

Another important remark in this optic is the fact that the geometries generated in the gbXML model doesn't correspond globally to the original object modelled in Revit, but they can be just portions that adjacent rooms come in contact to. As a example, if a wall is shared by two rooms, this wall will be exported as two separated surfaces for each room involved.

Correspondence with  
the original Revit object

The IFC classes generated, instead, all correspond entirely to a object modelled in Revit, preserving in this way the elements composition of the original model also in Cesium.

Another important observation involve the difference in the exportation manageability for the two formats. While the IFC conversion can be accurately customized and filtered through the exportation panel and it can be view's dependent, for the gbXML we have few options to imperceptibly modify the outputting geometric model. We can't chose which category we want to include and either select single surfaces to not be exported.

Customized exportation

The exportation to IFC can be managed object by object, and every physical instance in the Revit model can be exported: from the walls to the furnitures, from the structural elements to the mechanical. The planar surfaces which compose the gbXML model are instead automatically generated, as they are the bounding surfaces of the thermal zones, the elementary parts of the energetic model at the base of the gbXML. These planar elements are then organized in categories of surfaces (ExteriorWall, Roof, etc...) on the basis of their orientation towards the concerned thermal zone and their adjacency with the exterior or not. This automated generation, beside the fact that can't be manually managed, is also a continuous source of issues, like the failed generation due to a imprecise delimitation of the rooms or the assignment to a erroneous category of surfaces.

For what involve the exportation of the Revit families parameters, the dissertation doesn't even exist since in the gbXML exportation procedure there is no way to manage, either include, this kind of information. The only attributes exported are those expected by the gbXML schema, and we can't even use these "slots" to insert a particular attribute, since all the exported information in gbXML are metadata, that doesn't comes from a manually edited text field in Revit.

Properties exportation

In IFC, on the contrary, we can store all the information we want, adding to automatically generated parameters also personalized. During the exportation procedure, we have previously explained the possibility of organize and filter the information to be transferred as we made with the exportation of the project dedicated schedules as exclusive Property Sets.

gbXML's geometries  
ease of conversion

gbXML has however some advantages, deriving from its above mentioned "bottom-up" schema, which implies a simpler and easier job in terms of geometry parsing and conversion. Unlike most of the IFC's Shape Representations, which involve several reference system rotations and complex parameters of extrusion, addition, etc, the gbXML describes its planar geometries as a sequence of cartesian points which describes the vertex of these polygons in their X,Y,Z local coordinates.

IFC geometries conversion requires instead a case by case deepened analysis, considering that every shape representation and every class has a different methodological approach.

#### 4.1.2.2. BEST CASE CHOICE

In light of these characteristics typical of each exchange format considered, we can now take a position selecting the most suitable and performing configuration, whereby we can further develop this project.

gbXML only visualization

On the basis of its light and ease geometrical structure, gbXML could be an interesting way to represent BIM models on a GIS platform. But this must be limited for a low detail visualization, since its structure based on planar surfaces is inadequate for the visualization and interrogation of detailed building elements as for the present project.

Mixed visualization

We can consider the possibility of using the gbXML, maybe only for the medium detailed LoD2 visualization, combining its geometrical representation with the support of the properties deriving from the IFC exportation. This could be possible thanks to the univocal connections between gbXML elements and IFC entities described in "3.3.4. Connections between Revit model and exchange models". But beyond this opportunity, we have to consider the fact mentioned above that the gbXML elements doesn't describe effectively the original building model, but instead they are fragments of a energetic model.

The final decision: IFC  
only visualization

In the light of recorded performances at the end of this project, the most satisfying option is without a doubt the IFC only combination, where we can show geometries and attributes contained in this format. The issues recorded must be so implemented in order to make this visualization more and more complete.



## 4.2. METHODOLOGY

Once we have analysed the final output of the procedure, we can make assessments about the methodological approach, evaluating how the progressive stages and the decisions made approached us step by step to the final target, in terms of integration of the BIM model in the GIS environment.

The first assessment is about the standardised operations involved across the methodological process.

Importance of the  
standardisation of the  
process

The definition of a standardised nomenclature had an important role in the development of the work. As a reminder, this procedure has been faced in two different moments of the process:

- During the definition of the attributes to be shown LoD per LoD;
- During the definition of the Revit schedules.

In the first case, the standardisation was supported by the creation and then the use of Shared Parameter file containing all those attributes, so that we had to use them along the BIM phase, modeling the building and managing the property schedules. The result is a uniformity in the informations' content of every object placed into the model, avoiding in this way redundancy and lexical errors which could implicate errors during the parsing procedures.

In the second case, the naming of the Revit schedules had an important role also in terms of final visualization on Cesium. In fact, the specific encoding used allowed the conversion software to find in the IFC exchange model the properties attached to the different exported objects in a complete and univocal way. More precisely, as represented in Fig. 4–5, the software is implemented with a decoding tool that, once it finds in the model a `IfcPropertySet`, is able to analyse its "name" string attribute and recognize, in particular, in which visualization level this set of parameters must be shown. Once individuated and analysed the Property Set, the software can suddenly parse the `IfcSingleValue` contained inside and extrapolate from them their "Name" and "Value" attribute. In this way, the same object of the model can show different attributes according to the LoD represented on Cesium.

Standardisation has involved also the definition of specific export setups, with the creation and saving of ready to use conversion settings file, that can be shared and included by different users. This implicates a wide uniformity of the exported models and reduces the presence of issues due to individual errors.

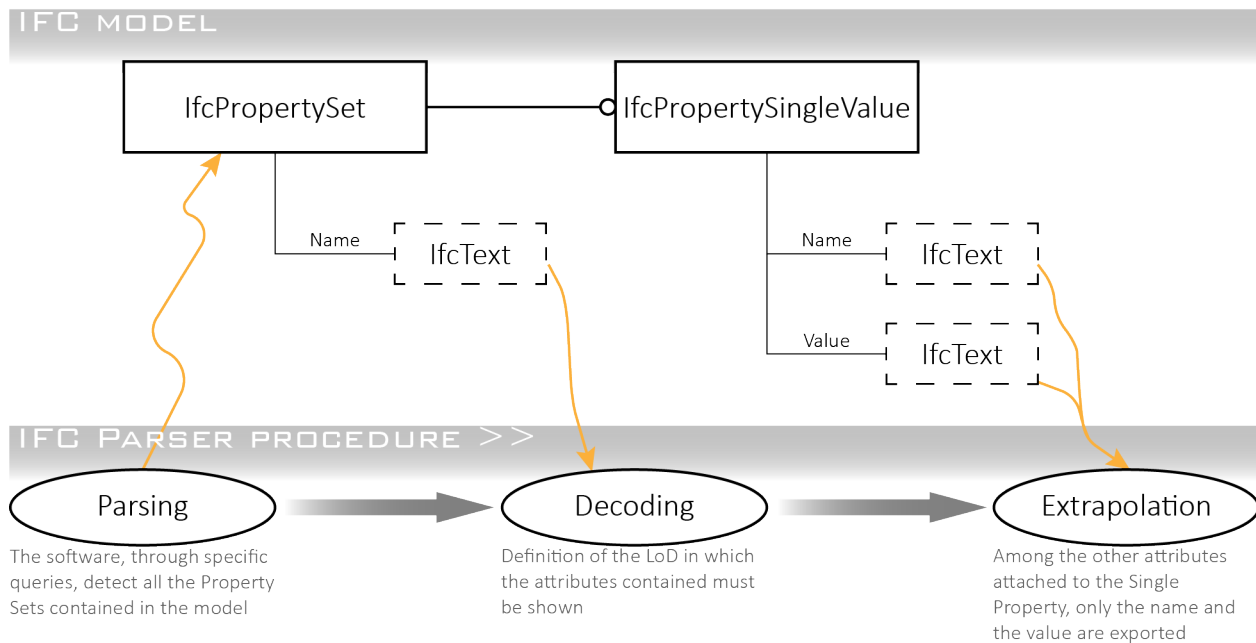


Fig. 4–5 - Nomenclature decoding

#### Georeferencing and rotation of the Revit model

The preliminary operations of georeferencing and rotation of the Revit model, described in "3.3.1.3. Georeferencing and rotation of the model", had also satisfying results at the end of the process. Those procedures gave in fact to the BIM model a connotation which allows it to communicate with the GIS technology and to be loaded automatically in the Cesium platform.

The coordinates obtained and then attached to a specific Survey Point in the Revit model, are in fact exported both in the gbXML model both in the IFC.

Here below, they are shown as they are exported in the gbXML format, as a 6 characters based geographic coordinates, inside the Location element.

```

<Campus id="aim0002">
  <Location>
    <StationId IDType="WMO">160478_2006</StationId>
    <ZipcodeOrPostalCode>00000</ZipcodeOrPostalCode>
    <Longitude>7.76864</Longitude>
    <Latitude>45.1382</Latitude>
    <Elevation>709.8792</Elevation>
    <CADModelAzimuth>0</CADModelAzimuth>
    <Name>Settimo Torinese, Piem., Italy</Name>
  </Location>
  ...
</Campus>
  
```

In the IFC model, the IfcSite class contains, in the RefLatitude and RefLongitude attributes, the same coordinates but in DMS ("Degree,minutes,seconds") coordinates.

```

#73000= IFCSITE('OmOudRzQbENhciOrpJDPJS',#41,'Default',$,$,#72999,$,$,ELEMENT,(
  45,7,59,484558),(7,46,5,758323),0,$,$);
  
```



In this way, the conversion software can extrapolate informations about the placement of the object in the globe and attach them to the B3DM file generated by the 3DTiles Generator.

Since the model is georeferenced, we can load it to the Cesium application without manually editing the coordinates. This aspect allow us to improve the automation of the process, since this procedure can be executed by the Middleware, without user control, making it an important advantage when several building models must be loaded as in a district scale project like this.

Another relevant observation is about the capability of the dashboard to perform a visualization based on Level of Detail. This result was one of the expectations which lead us to address the conversion output towards the glTF format and, above all, the 3DTiles streaming system.

Visualization based on  
Level of Details

This combination of tools creates a structure able to collect under the same batch different versions of the same model, characterized by increasing complexity of geometries, that are loaded separately on the virtual globe according to specific criteria of elevation and viewpoint.

This allow to not have a static representation of the scene, gaining instead a customizable "smart" 3D streaming where we can experience great advantages in terms of rendering speed and fruition and interrogation of the represented objects and properties attached to them.



## 5. CONCLUSIONS

## 5.1. FINAL ASSESSMENTS

The final assessments can be made on two levels. The first focused on the outcomes showed in the "4.1. Visualization" paragraph; the second oriented towards an analysis of the methodological approach adopted during the project and studied in its final results in "4.2. Methodology".

### Assessment on the results in the visualization

If we observe the outcomes in terms of final visualization we have to notice a evident discrepancy between the attended results supposed in "2.2.3. Attended results" and what we finally got.

The DIM dashboard has in fact a more primitive interface as the expected one, the navigation is limited to the main window and it hasn't been developed the tree structure panel for selecting and show/hiding the elements. Geometries aren't well exported and many entities aren't yet supported in the visualization. We can assert that, in this case, we are in a deep "work in progress" situation, with so many issues to be faced and implemented.

### Assessment on the results in the methodology

But, more than the actual and immediate results, it's more important to concentrate on the methodological approach and compare the results obtained and showed in "4.2. Methodology" with the proposed objectives theorized in "2.2.2. Objectives' definition".

In this optic, we can confirm the previously listed satisfying results, because the whole interoperability and integration process started from the BIM model has found a way, from the starting point to the end, which gave us the expected outcomes.

Even if the apparent results in the DIM interface show many issues, the aspect we have to consider and take as a greater success is the fact that the methodology works. The geometries can be exported, converted and visualized. The properties are also parsed, gathered and then attached again to the original object. A way it has been found and now we must only blaze this trail and make it come a completely automated and integrated highway between the BIM and the GIS worlds.

In the wake of this kind of assessment, it's possible to make a wider discussion, facing those aspects that could be the future developments.

## 5.2. FUTURE DEVELOPMENTS

This paragraph gain great perspectives of development thanks to the theoretical basis of the proposed project, which has been structured around the concepts of "open" and "integration". These concepts gave to the project a dynamic and borderless structure, that can draw inspiration from the limitless universe of the open source universe.

The philosophy at the base of the open source is that everything must be shared and, adopting it, we'll be allowed to not be constrained by a software vendor, which will limit our possibilities and creativity.

The development of a process based on this premises, allows to draw a methodology and a software able to satisfy precisely and accurately our needs: if user and developer collaborate the result is something that will clearly please both. But the development doesn't start from nothing, is instead a customization and implementation of an existing product or specification, made using libraries and tools available in the free market.

Also in terms of interoperability, which is a milestone of this project, we must notice that open source softwares are much better at adhering to open standards than proprietary softwares are.

According to these premises, we can assume that the future developments of this project can't be limited to the reaching of the proposed objectives, but it can and must be implemented for greater performance and utilities.

Once the connection between BIM and GIS is resolved also in terms of visualization, the DIM platform must be implemented with contributes of energy analysis tools for simulations and ICT, in order to apply it to the same purposes of projects like DIMMER and EeB.



# ACKNOWLEDGEMENTS





I would like to express my deep gratitude to Professor Anna Osello, my research supervisors, for the great opportunity she gave me to start this stimulating search path, from the stage experience till the development of this thesis.

I would also like to thank Matteo del Giudice and Francesca Ugliotti, my tutors and supervisors of this research, for their great willingness and extraordinary support showed during this thesis process.

This work would not have been possible without the fundamental support from Fabrizio Massara, which accompanied me throughout this work, contributing with immense dedication and efforts to this thesis.

I would like to thank also my friends who supported me during this laborious times, encouraging me and letting me pass wonderful moments of happiness.

Finally, I wish to thank from the bottom of my heart my parents for their support and encouragement, and their love daily demonstrated to me throughout my study years.



## BIBLIOGRAPHY

- [1] Kang T. W., Hong C. H., "A study on software architecture for effective BIM/GIS-based facility management data integration", March 2015;
- [2] Karan P., Irizarry J., Haymaker J., "BIM and GIS Integration and Interoperability Based on Semantic Web Technology",
- [3] Trubka R., Glackin S., Lade O., Pettit C., "A web-based 3D visualisation and assessment system for urban precinct scenario modelling", December 2015;
- [4] Deng Y., Cheng C. P., Anumba C., "Mapping between BIM and 3D GIS in different levels of detail using schema mediation and instance comparison", July 2016;
- [5] Deng Y., Moumita D., "A cloud computing approach to partial exchange of bim models", 2013;
- [6] Brovelli M. A., Kilsedar C. E., Hogan P., Prestifilippo G., Zamboni G., "NASA World Wind virtual globe for an open smart city", March 2017;
- [7] Elvidgea C. D., Tuttle B. T., "How virtual globes are revolutionizing earth observation data access and integration", 2008;
- [8] Osello A., "District Information Modeling: implementation and standard definition", September 2016;
- [9] Succar B., "The Five Components of BIM Performance Measurement", January 2010;
- [10] Succar B., "Building Information Modelling Maturity Matrix", January 2010;
- [11] Succar B., "Building information modelling framework A research and delivery foundation for industry stakeholders", 10 October 2008;
- [12] Succar B., Kassem M., "Building Information Modelling: Point of Adoption", CIB World Congress, Tampere Finland, May 30- June 3, 2016;
- [13] Chamila D. D. Ramanayakaa, Senthilkumar Venkatachalamb, "Reflection on BIM development practices at the pre-maturity" in *Procedia Engineering* 123 ( 2015), 462 – 470;
- [14] British Standards Institution, *PAS 1192-2:2013 Incorporating Corrigendum No. 1 - Specification for information management for the capital/delivery phase of construction projects using building information modelling*, February 2013;
- [15] Haron A. T., Marshall-Ponting A. J., Aouad G., "Building information modelling: Literature review on model to determine the level of uptake by the organisation", 2010;
- [16] Cabinet Office, *Government Construction Strategy*, July 2012;

- [17] EDSL, *EDSL Guide for Revit gbXML Files*;
- [18] Karl-Heinz Häfele, *IFC and gbXML, 2 Building Information Models for Building Performance Simulation*, Institut für Angewandte Informatik, Karlsruher Institut für Technologie;
- [19] *Introduzione a XML*, Università di Bologna;
- [20] Lagüela S., Díaz-Vilariño L., Martínez, J. Armesto J., " Automatic thermographic and RGB texture of as-built BIM for energy rehabilitation purposes", 14 December 2012;
- [21] Sokolov I., John Crosby J., *Utilizing gbXML with AECOsim Building Designer and speedikon - Building Performance Analysis Using Bentley Products*, October 2011;
- [22] *DesignBuilder Revit – gbXML Tutorial*, Design Builder Software;
- [23] Jianping Zhang, Fangqiang Yu & Ding Li, Zhenzhong Hu, " Development and Implementation of an IndustryFoundation Classes-Based Graphic Information Model for Virtual Construction" in *Computer-Aided Civil and Infrastructure Engineering* 29 (2014), 60 – 74;
- [24] Stanford University, *Data Modelling Using EXPRESS-G for IFC Development*;
- [25] Robert W. Schuler, "The Application of ISO 10303-11 (the EXPRESS Language) in Defining Data Models for Software Design and Implementation", April 2001;
- [26] Kull A., *Compatibility issues with BIM*, Royal Institute of Technology (KTH) Department of Civil and Architectural Engineering, Stockholm, Sweden, 2012;
- [27] Liebich T., *IFC 2x Edition 3: Model Implementation Guide*, May 18, 2009;
- [28] British Standards Institution, *ISO 10303-11 :1994 - Industrial automation systems and integration - Product data representation and exchange. Part 11.Description methods: the EXPRESS language reference manual*, 1994;
- [29] Kiviniemi A., Tarandi V., Karlshøj J., Bell H., Karud O. J., *Review of the Development and Implementation of IFC compatible BIM*, Erabuild, 2008;
- [30] Wix J., *What is IFC?*, AEC 3, Building Smart;
- [31] Dong B., Lam K. P., Huang Y. C., Dobbs G. M., "A comparative study of the IFC and gbXML informational infrastructures for data exchange in computational design support environments", Carnegie Mellon University, 2007;
- [32] Hyunjoo K., Zhenhua S., Inhan K., Karam K., Annette S., Jungho Y., "BIM IFC information mapping to building energy analysis (BEA) model with manually extended material information", 3 April 2016;
- [33] Pinkos H, *Migrating from Google Earth to Cesium*;
- [34] Chaturvedi K., "Web based 3D analysis and visualization using HTML5 and WebGL", March

2014;

[35] Keyzers J., *Review of digital globes 2015*, March 2015;

[36] Dong B., Lam K. P., Huang Y. C., Dobbs G. M, "A comparative study of the IFC and gbXML informational infrastructures for data exchange in computational design support environments", 2007;



## WEBSITES<sup>1</sup> REFERENCES

- [37] [https://www.graphisoft.com/it/soluzioni/open\\_bim/](https://www.graphisoft.com/it/soluzioni/open_bim/);
- [38] <https://www.thenbs.com/knowledge/bim-levels-explained>;
- [39] <http://www.bimthinkspace.com/2009/12/episode-13-the-bim-maturity-index.html>;
- [40] <http://www.bimthinkspace.com/bim-maturity/>;
- [41] <https://github.com/KhronosGroup/glTF/blob/master/specification/2.0/README.md>;
- [42] <http://www.json.org/json-it.html>;
- [43] <https://www.khronos.org/glTF>;
- [44] <http://www.dimmerproject.eu/>;
- [45] <http://www.drawingtothefuture.polito.it/projects/seempubs/>;
- [46] <http://www.oracle.com/technetwork/java/javasee/tech/index.html>;





# ANNEXES


## A. GBXML GEOMETRIES' REPRESENTATION

In this attachment it will be analysed the exportation gbXML file of a Revit room, in order to analyse how the geometries are described in the XML structure of this format.

In particular we will focus on the X, Y, Z coordinates used by the gbXML "CartesianPoint" attribute to describe the different kind of geometries generated, and we will each time represent the vertex in question with an inherent 3D representation.

As described during the dissertation, the gbXML coordinates are local coordinates, relative to a 0, 0, 0 point which derives from the Revit Survey Point geographic placement. These geographic coordinates are contained in the "location" element as it follows.

```
<Location>
  <StationId IDType="WMO">160478_2006</StationId>
  <ZipcodeOrPostalCode>00000</ZipcodeOrPostalCode>
  <Longitude>12.6</Longitude>
  <Latitude>41.8</Latitude>
  <Elevation>100.8888</Elevation>
  <CADModelAzimuth>0</CADModelAzimuth>
  <Name>Roma, Italia</Name>
</Location>
```

All the gbXML vertex coordinates are so expressed as meters of distance from this point which has been represented in the following images with the  Revit symbol.

### SPACE ELEMENT

The first geometry defined in the gbXML model is the "Space" element, the 3D global representation of the room, which is represented with 3 kinds of element:

- **1 PlanarGeometry:** which defines the planar imprint of the space;
- **1 ShellGeometry:** a list of cartesian point, that describes the vertex of the surfaces which defines the room;
- **6 SpaceBoundary:** each defining a surface of the 3D volume.

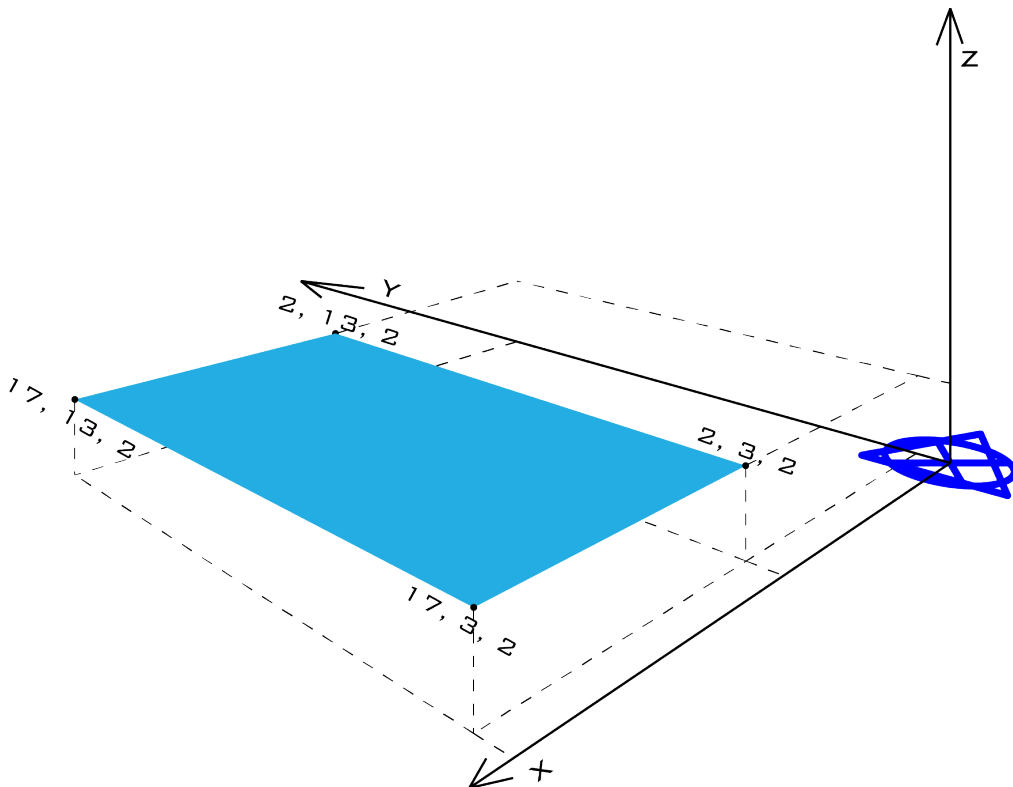
#### PLANAR GEOMETRY

```
<PlanarGeometry>
  <PolyLoop>
    <CartesianPoint>
      <Coordinate>2</Coordinate>
      <Coordinate>3</Coordinate>
```

```

<Coordinate>2</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>17</Coordinate>
  <Coordinate>3</Coordinate>
  <Coordinate>2</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>17</Coordinate>
  <Coordinate>13</Coordinate>
  <Coordinate>2</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>2</Coordinate>
  <Coordinate>13</Coordinate>
  <Coordinate>2</Coordinate>
</CartesianPoint>
</PolyLoop>
</PlanarGeometry>

```



## SHELL GEOMETRY

```

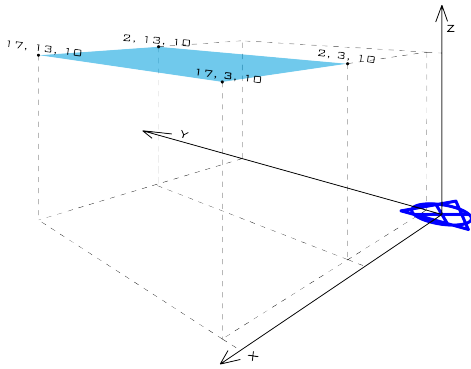
<ShellGeometry id="aim0034">
  <ClosedShell>
    <PolyLoop>
      <CartesianPoint>
        <Coordinate>2</Coordinate>
        <Coordinate>3</Coordinate>
        <Coordinate>10</Coordinate>
      </CartesianPoint>
      <CartesianPoint>
        <Coordinate>17</Coordinate>
        <Coordinate>3</Coordinate>
        <Coordinate>10</Coordinate>

```

```

</CartesianPoint>
<CartesianPoint>
  <Coordinate>17</Coordinate>
  <Coordinate>13</Coordinate>
  <Coordinate>10</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>2</Coordinate>
  <Coordinate>13</Coordinate>
  <Coordinate>10</Coordinate>
</CartesianPoint>
</PolyLoop>

```



The above schematized surface is the first of the 6 polyloop that defines the ClosedShell element, in other words, the bounding box of the room.

Suddenly are reported the other polyloops and finally it will be represented the ClosedShell entirely.

```

<PolyLoop>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
</PolyLoop>
<PolyLoop>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>10</Coordinate>
  </CartesianPoint>

```

```

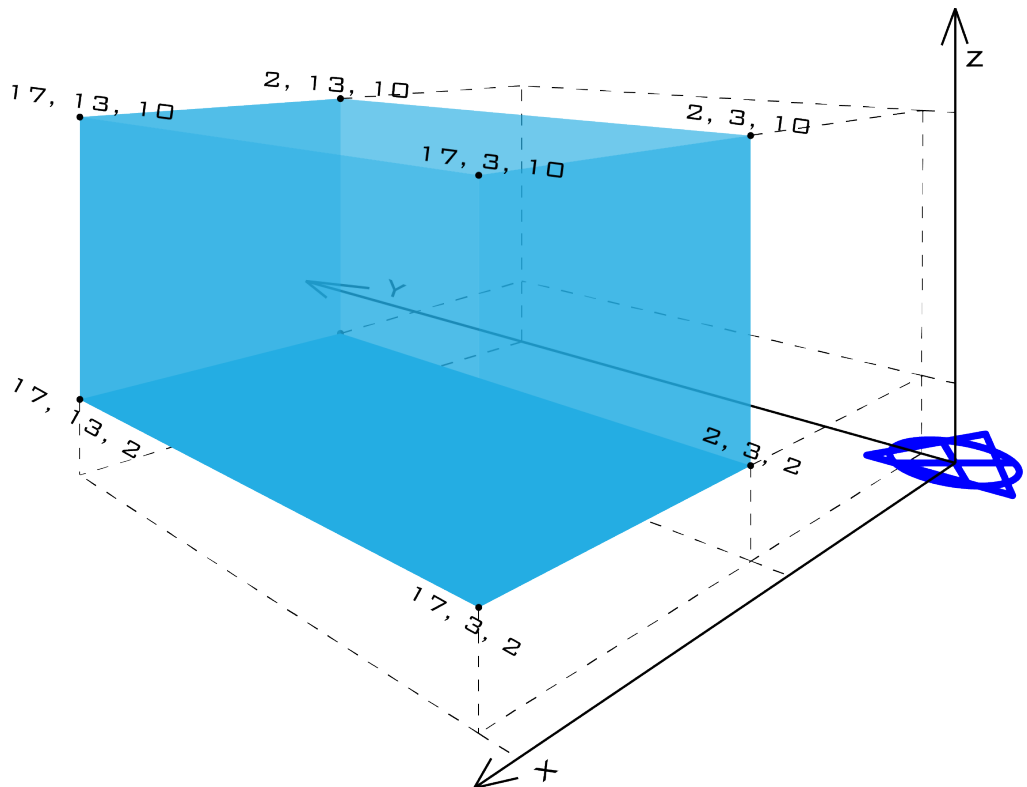
</CartesianPoint>
<CartesianPoint>
  <Coordinate>17</Coordinate>
  <Coordinate>3</Coordinate>
  <Coordinate>10</Coordinate>
</CartesianPoint>
</PolyLoop>
<PolyLoop>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>10</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>10</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
</PolyLoop>
<PolyLoop>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>10</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>10</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>13</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
</PolyLoop>
<PolyLoop>
  <CartesianPoint>
    <Coordinate>2</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>
  <CartesianPoint>
    <Coordinate>17</Coordinate>
    <Coordinate>3</Coordinate>
    <Coordinate>2</Coordinate>
  </CartesianPoint>

```

```

</CartesianPoint>
<CartesianPoint>
  <Coordinate>17</Coordinate>
  <Coordinate>3</Coordinate>
  <Coordinate>10</Coordinate>
</CartesianPoint>
<CartesianPoint>
  <Coordinate>2</Coordinate>
  <Coordinate>3</Coordinate>
  <Coordinate>10</Coordinate>
</CartesianPoint>
</PolyLoop>
</ClosedShell>
</ShellGeometry>

```



### SPACE BOUNDARY

```

<SpaceBoundary isSecondLevelBoundary="false" surfacIdRef="aim0086">
  <PlanarGeometry>
    <PolyLoop>
      <CartesianPoint>
        <Coordinate>17</Coordinate>
        <Coordinate>3</Coordinate>
        <Coordinate>2</Coordinate>
      </CartesianPoint>
      <CartesianPoint>
        <Coordinate>17</Coordinate>
        <Coordinate>13</Coordinate>
        <Coordinate>2</Coordinate>
      </CartesianPoint>
      <CartesianPoint>
        <Coordinate>17</Coordinate>
        <Coordinate>13</Coordinate>
        <Coordinate>10</Coordinate>

```

```

    </CartesianPoint>
    <CartesianPoint>
      <Coordinate>17</Coordinate>
      <Coordinate>3</Coordinate>
      <Coordinate>10</Coordinate>
    </CartesianPoint>
  </PolyLoop>
</PlanarGeometry>
</SpaceBoundary>
<SpaceBoundary isSecondLevelBoundary="false" surfaceIdRef="aim0097">
  ...
</SpaceBoundary>
....

```

## SURFACE ELEMENTS

These elements describe single planar geometries, so they correspond to the 6 polygons which close the room. But, unlike the SpaceBoundary elements, the Surfaces have an analytical and physical connotation, since they can be related to a SurfaceType attribute and to Construction elements.

Here is reported the XML schema of one surface, a ExteriorWall.

```

<Surface surfaceType="ExteriorWall" exposedToSun="true" id="aim0108">
  <AdjacentSpaceId spaceIdRef="aim0024" />
  <RectangularGeometry id="aim0109">
    <Azimuth>270</Azimuth>
    <CartesianPoint>
      <Coordinate>2</Coordinate>
      <Coordinate>13</Coordinate>
      <Coordinate>2</Coordinate>
    </CartesianPoint>
    <Tilt>90</Tilt>
    <Width>10</Width>
    <Height>8</Height>
  </RectangularGeometry>
  <PlanarGeometry>
    <PolyLoop>
      <CartesianPoint>
        <Coordinate>2</Coordinate>
        <Coordinate>3</Coordinate>
        <Coordinate>2</Coordinate>
      </CartesianPoint>
      <CartesianPoint>
        <Coordinate>2</Coordinate>
        <Coordinate>3</Coordinate>
        <Coordinate>10</Coordinate>
      </CartesianPoint>
      <CartesianPoint>
        <Coordinate>2</Coordinate>
        <Coordinate>13</Coordinate>
        <Coordinate>10</Coordinate>
      </CartesianPoint>
      <CartesianPoint>
        <Coordinate>2</Coordinate>
        <Coordinate>13</Coordinate>
        <Coordinate>2</Coordinate>
      </CartesianPoint>
    </PolyLoop>
  </PlanarGeometry>

```

```
</PlanarGeometry>  
<CADObjectId>Muro di base: Generico - 30 cm [128876]</CADObjectId>  
<Name>W-1-E-W-3</Name>  
</Surface>
```