INTEGRATING BUILDING-LEVEL AND CAMPUS-LEVEL DATA

by

Lan Wei

B.C.S., The University of British Columbia, 2008

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

September 2013

Abstract

Integrating building design data (in the form of Building Information Models - BIMs) and 3D City Models is a promising solution to retrieving both detailed building information and exploring the relationships among building components and buildings in a city area. It could also help to facilitate decision making in building facility management and maintenance operations. Among other challenges, performing such an integration is difficult because BIMs and 3D city models vary in both semantic and geometric representations.

Our first attempt was to try to convert the BIM into the city model, but the results were very disappointing. It is infeasible to convert every necessary component in the BIM into the city model, and the converted files cannot be visualized by the current city model applications. Next, we implemented a data integration system to incorporate information from both models. We used a novel approach to apply arithmetic expressions to express the overlapping information, not only for the semantic representation, but also on the geometric representation of the building components. This approach improved query answering on components from both models. Finally, we describe future challenges that will be needed to improve the accuracy of our current approach.

Preface

My supervisor Dr. Rachel Pottinger identified this research topic for me and she also guided me through my entire project by pointing me the right directions and possible solutions.

In Section 4.1, Arianne Dee generated an FME conversion mechanism to convert IFC files into CityGML. I used that to compare with the results that BIMServer created. In addition, Arianne created a diagram displaying the approaches of transforming building models into CityGML models. I built upon this diagram to create Figure 5.

Other than the contributions listed above, this thesis is original, unpublished, independent work by the author, L. Wei

Table of Contents

List of Tables

List of Figures

Acknowledgements

I particularly appreciate my supervisor Dr. Rachel Pottinger, who inspired me to overcome the challenges in my research and provided considerate support.

I also thank Dr. George Tsiknis for willing to be my second reader and providing valuable suggestions on my thesis.

Special thanks are owed to my family, who generously supported me during my study.

Chapter 1
Introduction

The increasing demand on facility management, inspection, and maintenance operations has caused an urgent need to integrate building information models and geographic information systems models. Consider the following real world example: someone made a request for maintenance to look at the temperature in "core chem Physics office A249", since it was too cold (14 degrees). The operators need to use a campus building operation system to locate the particular building and room cited in the work request. Then they check the heating and cooling systems that serve that particular area or room. In addition to the location of the equipment (such as the heating system) in the building, the maintenance personnel may need information about the manufacturer, serial number, maintenance history information, service manual, and spare part information about the specific equipment that needs to be maintained or repaired or replaced. In order to create such a comprehensive campus operation system to retrieve both building information and explore the relationships among buildings in a campus area, we need the system to contain both the geospatial campus data and a maximum level of detail about the real buildings. Currently, we do not have any single model or system that incorporates both of them; they exist only in separate models. Therefore, in order to perform the above maintenance request, we need to build an integrated system which combines all the needed information.

There are many different building information models and 3D geospatial information models representing real world surrounding objects from both geometrical and semantic perspectives. Today, Industry Foundation Classes (IFC) and City Geography Markup Language (CityGML) are two of the most prominent semantic models; IFC is used for design, and CityGML is used for describing built building infrastructures [1]. IFC [2] was introduced as a standard for describing building components and construction data. CityGML [3] is a common information model and XML-based encoding system for the representation, storage, and exchange of virtual 3D city and landscape models. The integration of IFC and CityGML is seen as a necessary step for getting a complete 3D city model with detailed building information. The reason for this is that this integration would assist us in handling the operation and maintenance requests by exploring the 3D campus view to locate the desired buildings (using CityGML) and zoom into the building to display the layout of every floor so that we can easy identify the room that stores the equipment that needs to be checked and/or repaired (using IFC).

Generally speaking, all the previous integration approaches to the implementation of a semantic and geometrical integration of IFC and CityGML models [4][5][6] have focused mainly on unidirectional conversion from IFC to CityGML, because CityGML is more popular for decision-making as it has a broader view. They define some standard mapping schemas between IFC and CityGML based on the semantic agreement of their interpretation of the same objects. There are conversion tools [6] which extend CityGML with rich semantic information of IFC and ADE. In addition, commercial software products for conversion from IFC to CityGML, such as IfcExplorer [7] and Safe software [8], make a great effort to the development of 3D city modeling integration.

Our goal in this thesis was to build an operation and maintenance system to allow users explore and retrieve a wide range of campus data, starting from campus scale and zooming into a room in a building. To allow this integration, we need to be able to simultaneously query IFC and CityGML. CityGML represents geospatial environment as well as the building components. On the other hand, IFC only has the detailed building information, excluding the outer surrounding environment. After considering the previous conversion approaches [4][5][6][9] and commercial software products [7][8][8][10], we chose to build our integration system on top of the existing work. We selected two most suitable frameworks as a start point of our conversion from IFC to CityGML, BIMServer [10] and Feature Manipulation Engine (FME) [11].

After comparing and analyzing the converted results from BIMServer and FME, we decided to extend the CityGML schema to accommodate the rich semantic building information that the existing conversion tools cannot cover. After running several experiments, we concluded that it is not currently possible to bring all the details for buildings such as the mechanical components and facility information into CityGML. We encountered a number of problems in our attempt: (1) the standard CityGML schema cannot currently support such kinds of information , and (2) file size increases dramatically when small amounts of additional information is added in. The file size is far beyond the maximum capability that the current conversion applications can handle.

After we realized that the conversion from one model to another model, no matter what direction it is, is infeasible to implement, we switched our approach to incorporate them together into one central system. There are two common systems to combine different data sources into one central destination, data integration and data warehouse. After thoughtfully considering their difference and respective usages, we decided to implement a data integration system [12] to incorporate IFC and CityGML schemas based on their overlapping information, which enables us to retrieve information from both of them on a simple and uniformed interface. We will explain its architecture and detailed implementation procedure in the late sections.

In the remainder of this thesis, Chapter 2 explains the terminologies and concepts used in our project, such as the IFC, CityGML, the schema in different standard formats, for instance Extensible Markup Language (XML) and relational model. Chapter 3 introduces the previous and ongoing research work and commercial products of conversion frameworks and tools. In Chapter 4, we introduce data integration and data warehouse respectively, make a comparison of their strengths and weaknesses in terms of storage size and construction time and explain the best circumstances they apply individually. Chapter 5 and Chapter 6 demonstrate the process we extract the IFC schema and CityGML schema in relational format from their standard schema specifications respectively. Next, we explain the steps of the process of establishing our data integration system and the challenges remaining in our project in Chapter 7. After that, we explore other ongoing research topics in the geographic information field and make a conclusion in the last chapter.

Chapter 2
Background

As we discussed above, to fully handle maintenance and operation requests, we need to access a complete picture of 3D city modeling at high levels of detail. We review the Building Information Model (BIM) [13] in detail in Section 2.1 and then discuss the higher-level CityGML [14] in Section 2.2.

2.1 BIM and IFC

"Building Information Modeling (BIM) is a digital representation of physical and functional characteristics of buildings." [15] A BIM describes buildings with respect to their geometric and semantic properties. Generally, it is generated at the early stage of the building's lifecycle to facilitate the architects, civil engineers and stakeholders in planning, designing and constructing. It has the ability to organize huge volumes of data related to buildings, the semantic information of building parts and spatial relationships between them, and also supports sophisticated 3D visualization and manipulation. Unlike CAD models, which represent buildings as a collection of points and lines, the semantic information that BIMs carry makes great contributions to the data analysis and decision making regard to buildings. It is an object oriented building modeling, which defines building components as elements and their properties and the relationships between them.

One of the most developed and established semantic models that implement BIM concepts is the Industry Foundation Classes (IFC) [2]. As an open standard schema, IFC is popularly used to exchange and share BIM data between different applications. Its standard schema comprises information contributing to a building's whole lifecycle: from conception, through design, construction and operation to maintenance and destruction [16]. IFC is a conceptual model for buildings which represents building structural components and their relationships semantically as shown in the below Unified Modeling Language diagram in Figure 1. It describes the components of spatial objects as classes and different arrows means different relationships and associations between classes. IFC has an IfcBuilding class which consists of one or more IFCBuildingStoreys. In each

IFCBuildingStorey, there are several IfcSpaces instead of rooms. The building elements are walls, roofs, beams, columns, stairs, or openings, such as doors, windows etc.

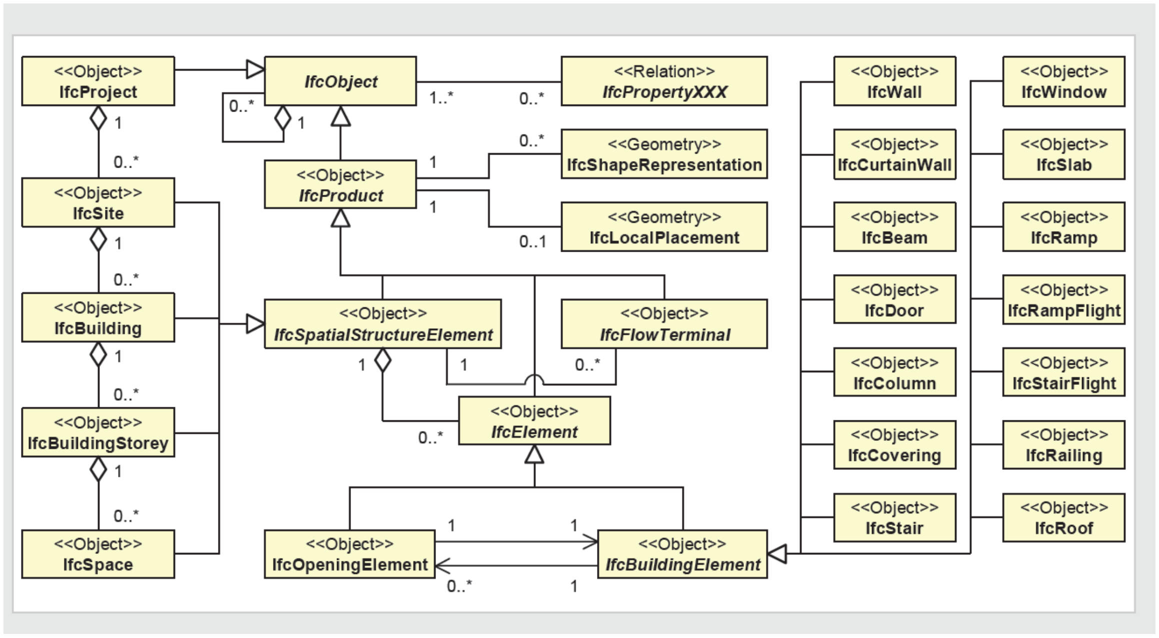


Figure 1: A UML notation of IFC building model.

2.2 CityGML

CityGML [3] is defined as a semantic information model, and it is used as a standard representation to store and exchange virtual 3D objects and city models among different applications. As a Geospatial (GIS) system, CityGML uses five different levels of detail (LoD) to represent the same city model objects in different degrees of detail, where objects become more detailed with the increasing LoD regarding both geometry and thematic differentiation:

- LoD0 represents the urban surfaces in which buildings are represented as footprints. As shown in the Figure 2, it is a topographic model of bare-earth terrain which displays the elevations of natural terrain features, such as barren ridge tops and river valleys [17]. However, elevations of the buildings and roads are digitally removed, so buildings are represented as 2D rectangle shapes as you can see from the top down.

- LoD1 is the first representation of buildings as 3D objects. All buildings look like blocks with the same flat roofs.
- In LoD2 and LoD3, a building is represented in an architectural model with details of roofs, walls, some exterior elements, such as balconies, and openings in the boundary surfaces, such as doors or windows. Lod3 applies more details and facade textures to the roofs and walls than LoD2, and it covers detailed vegetation and transportation objects which cannot be found in LoD2. [18]
- The most detailed level, LoD4 [18] adds detailed interior elements to buildings, such as rooms, interior doors, interior wall surfaces, stairs, furniture, electricity units, etc.

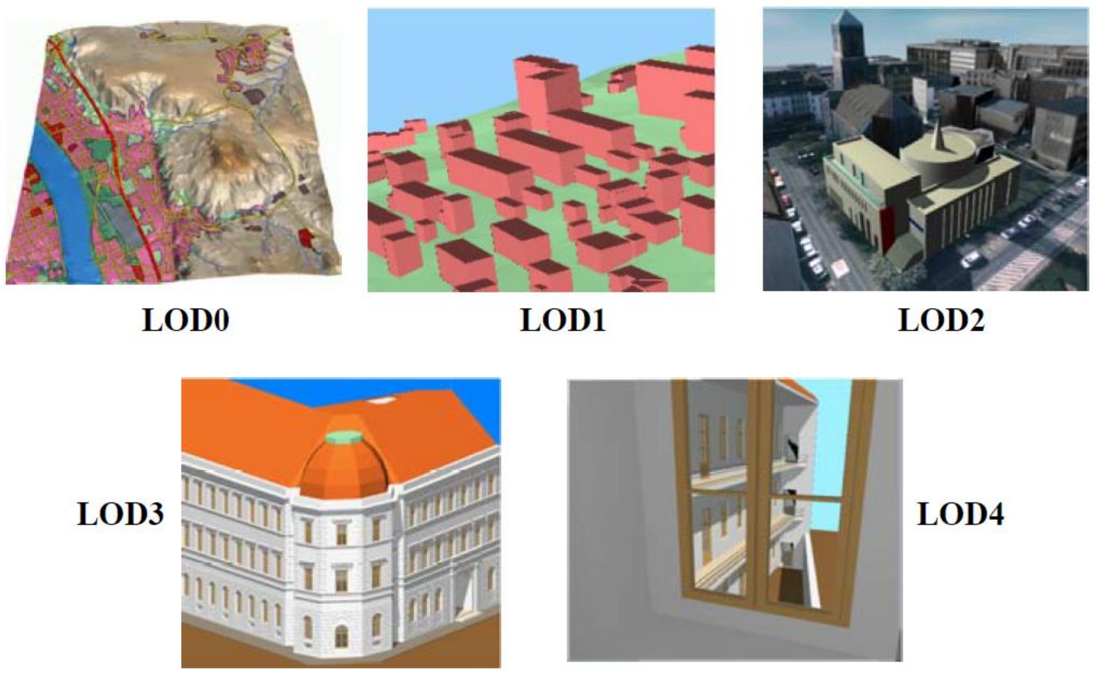
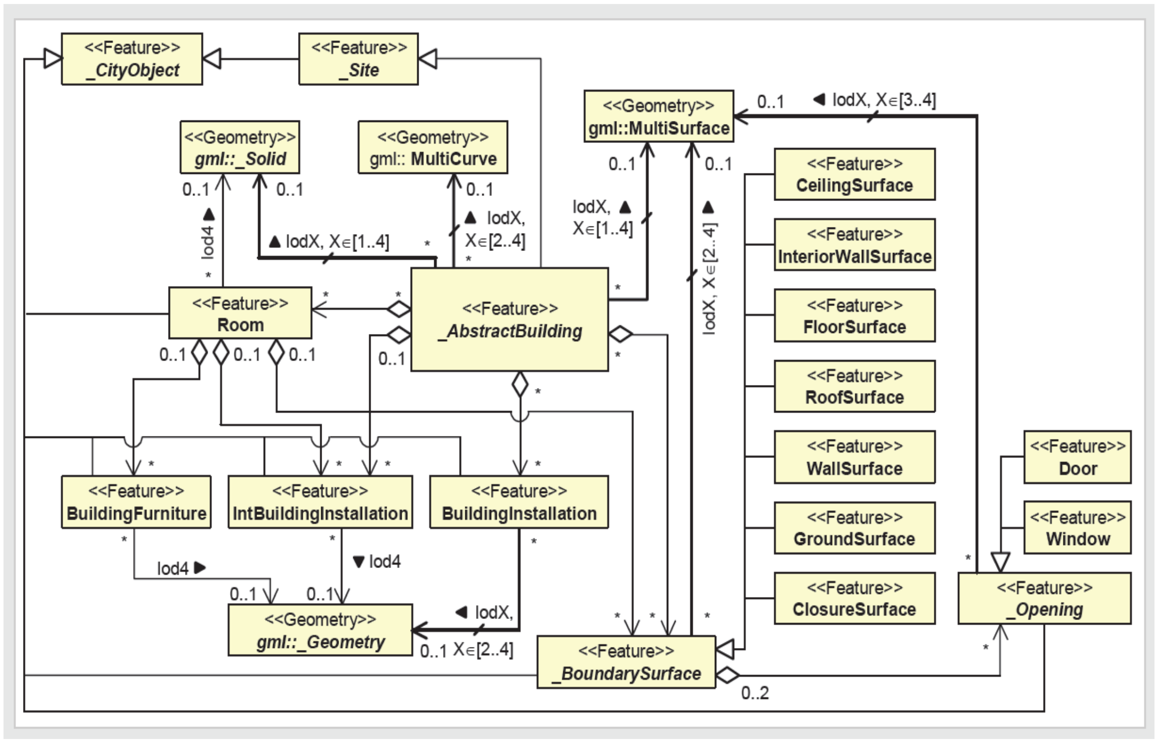


Figure 2: The five levels of detail (LOD) of CityGML (Source: IKG Uni Bonn [18]).

Thanks to these different levels of detail, CityGML is highly scalable, so the building details can be flexible in order to be adjusted to meet projects' special needs and in order to make efficient and sophisticated analysis. Compared to IFC, in Figure 3 CityGML does not have storey information, and it considers rooms as building components rather than spaces in IFC. CityGML represents the geographic information of spatial objects; thus, the measurement properties of each building component are represented as geographic coordinates instead of lengths and widths.

Generally, IFC [19], an element-based volume model, uses constructive solid geometry with volumetric and parametric primitives representing the structural components of buildings. However, CityGML, a surface model, uses boundary representation, which is the accumulation of observable surfaces of topographical features.



<<Feature>> _CityObject
<<Feature>> _Site
<<Geometry>> gml::MultiSurface
<<Geometry>> gml::_Solid
<<Geometry>> gml:: MultiCurve
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> RoofSurface
<<Feature>> WallSurface
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> Door
<<Feature>> Window
<<Feature>> _Opening
<<Feature>> Room
<<Feature>> _AbstractBuilding
<<Feature>> BuildingFurniture
<<Feature>> IntBuildingInstallation
<<Feature>> BuildingInstallation
<<Geometry>> gml::_Geometry
<<Feature>> _BoundarySurface
lod4
lodX, X∈[1..4]
lodX, X∈[2..4]
lodX, X∈[3..4]
0..1
0..2

Figure 3: A UML notation of CityGML Model.

Right now, FME Data Inspector supports the visualization of the IFC and CityGML files. The Solibri Model Checker provides sophisticated functions to analyze the BIM in the IFC format. Also, Galdos GML Inspector and LandXplore can view the CityGML data.

2.3 Basic database concepts and definitions

“A **schema** is a collection of entities (or classes), attributes, and relationships between entities.” [20] A schema defines the logical structure of the database; its formal definition language depends on the type of the database system. A relational database schema consists of a set of relations. Each relation represents an entity or a relationship between entities with a set of attributes to describe it. A relation is expressed in a format of

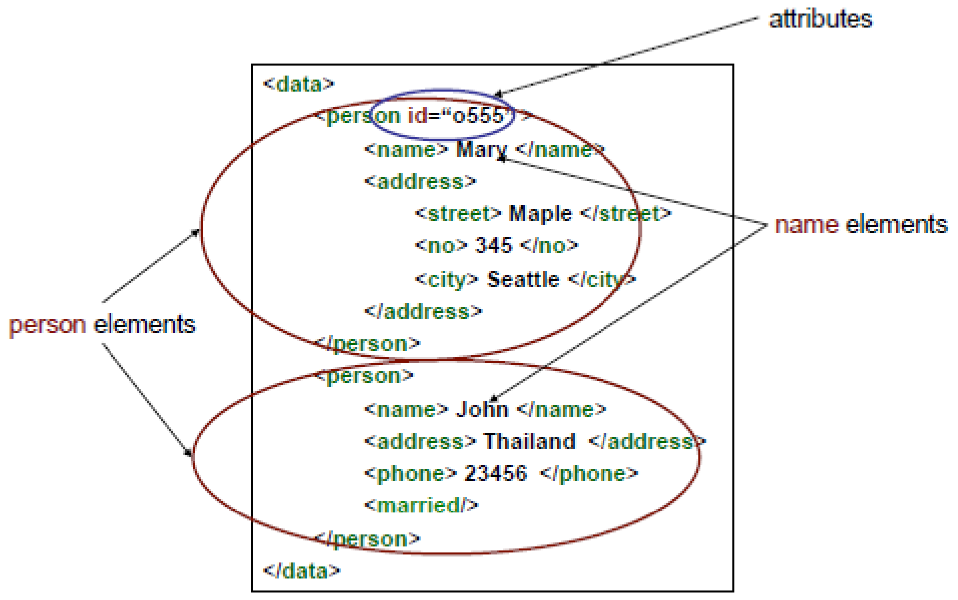
R(Attr$_1$, . . . , Attr$_n$), in which R is the name of the relation, Attr$_x$ is an attribute at the position x and the relation has n attributes.

"A **model** is the population of a schema, following the patterns, templates and constraints stipulated by the schema. It contains the actual instances of the entities (or classes)" [20]. Therefore, IFC and CityGML's standard specifications are schemas, but the IFC and CityGML files describing the real spatial objects are models.

Datalog is a declarative logic programming language, which has a format of head :- subgoal1, subgoal2, …, subgoal…. It has two parts separated by ":-", the left part is the head, and the right part is the body that consists of a group of subgoals. The head and subgoals are all relations, each of which has a name and a set of variables associated with it. The subgoals in the body can be placed in any order. Datalog is always used to declare queries, because the variables in the head can be expressed from the body, and it supports the basic query operations of select, project, and join. This is also called a conjunctive query. If the same variable name appears at multiple places in the body, it is a joining variable used to link several tables together.

2.4 XML

Unlike relational databases with fairly well-structured forms as tables, Extensible Markup Language (XML) is "a very popular standard format which [is being] used widely right now and…is a simple very flexible text format [that] is playing an increasingly important role in the exchange of a wide variety of data on the web and elsewhere" [21]. XML Schema Definition (XSD) is a standard language used to define XML schema, which describes the XML document structure and content in a brief and simple way. XML has a flexible structure, called semi-structured, because, while some of its content can come from part of the schema, it can express other information not in the schema. Compared to the relational schema, in which a relation is a set of attributes used to describe a concept or a relation in parallel, XML allows users to store heterogeneous data, in which relations can be hierarchically structured, while also allowing multiple values for one element. Therefore, its semi-structured format contributes to its expressiveness and flexibility, which not only facilitates data exchanges among different formats, but also allows for the storage of rich and diverse information.



```
<data>
  <person id="o555">
        <name> Mary </name>
        <address>
              <street> Maple </street>
              <no> 345 </no>
              <city> Seattle </city>
        </address>
  </person>
  <person>
        <name> John </name>
        <address> Thailand </address>
        <phone> 23456 </phone>
        <married/>
  </person>
</data>
```

Figure 4: Part of the XML text document from [22].

Figure 4 displays part of an XML text document of two people's names, addresses and phone numbers. The element names are enclosed within tags, such as "<person>", and the values are set between element tags, like "Mary". The heterogeneous structure allows for a hierarchy to be nested inside one element, such as, for example, the "street", "no", "city" elements are nested within the element "address", which belongs to the "person" element. Since it is not necessary that every person has a phone number, the address format can vary for different people.

On the other hand, because of XML's loose structure; many traditional database techniques cannot be applied to it to improve its performance, such as an index, for example. Since it lacks the classic query optimization, its syntax-parse based query execution algorithm is much less efficient than a relational database. Unlike the relational database, in which the data is stored in tables, XML is not intended for human beings to read and its redundant tag representation costs overwhelming storage waste. Additionally, it is difficult to detect improper format and/or syntax errors.

Chapter 3
Existing conversion approaches

In order to process the maintenance requests, such as the previous example's uncomfortable room temperature in the "core chem Physics office A249", we need to show not only the map of the entire campus, which indicates the paths to locate that building, but also the details of each floor in the specified building, so the maintenance worker can access the room where the heating system is located. While building the GIS model of the whole campus is considered a good starting point, we can then increase the picture by including the detailed building information from the BIMs. In sum, for all the operation requests, the most important thing we should do is to define the scale and region that we will operate at. We believe the best solution would be the integration of the BIM and the GIS to present the behavior of the entire campus and also the details of each individual building.

"Researchers all agree that the best approach for the integration of BIM and GIS is harmonized semantics" [23], which creates formal mappings between the detailed inner building models and outdoor real world GIS. Both the IFC and CityGML use different terminologies to describe the same domain. Semantic interoperability ensures that both IFC and CityGML share the same meaning for a defined spatial object. All the previous approaches are based on the idea of harmonized semantics; some of them focus on a unidirectional method (mostly from IFC to CityGML) for the conversion process. While part of this attempt is to develop an automatic framework to transform the IFC building models into CityGML with regard to each level of detail, another is to convert building information to GIS from CAD instead of BIM. Bidirectional conversion has recently drawn our attention as a means of fully integrating IFC and CityGML in terms of a conceptual model.

Some of the most significant existing approaches to the integration of IFC and CityGML are listed below:

The IFG Project

The framework of the IFC for the GIS (IFG) project (in 2003) [24] was designed to exchange building information between CAD systems and GIS, using IFC, so that it could support the analysis of the relationships among a building's different areas or other buildings, by accessing to the details of both the buildings and their surrounding environments. The project succeeded in creating mapping specifications from the IFC geometry to the GIS, and vice versa, by identifying entities in the IFC schema that could support the GIS application and creating mappings between them and the real GIS entities.

Nagel's framework of conversion of IFC to CityGML

In 2007, Nagel proposed a framework [4] that aimed at developing algorithms that automatically transform IFC building models into CityGML models, through a series of steps that create the separate footprints of each storey within their own boundary surfaces and finally merge them together. As this research only focused on LOD1 and LOD2 of CityGML, the purpose of the algorithms was to create a geometrically and semantically valid representation of LOD1, which could also be applied to LOD2.

Isikdag and Zlatanova's advanced framework

Isikdag and Zlatanova [5] have extended Nagel's framework by proposing a framework for automatic generation of building semantics and components in CityGML from their BIM representation. Because the CityGML and IFC models are designed for two different domains, they have very diverse objects and classes and cannot be directly and easily mapped to each other. Therefore, they generate semantic and geometric mappings for each LoD of CityGML separately, since the same object in one schema can be mapped to different objects in the other for different levels of detail. In order to simplify and facilitate the conversion process, for each level of detail, all the objects and attributes in both schemas that need to be transformed are first defined, as are the mapping rules.

A 3D Conversion Framework by Thomas Kolbe

With the aim of creating a holistic view of a 3D city, a team led by Thomas Kolbe at the Technical University of Berlin [25] proposed a framework that incorporates semantic spatial context data into 3D graphics/data of buildings and urban areas stored in formats such as X3D, DXF, KML, and COLLADA. The reason we chose CityGML as the intermediate layer for the conversion process to IFC is because it is a very rich semantic model that shares a similar notion of building semantics with the IFC. Therefore, we considered it a good bridge to link 3D visualised models to the sophisticated and detailed building models. The simple summary of this conversion framework is shown below: Geometric/graphics Models → Semantic City Model → Building Information Model. Their corresponding formats are:
X3D, DXF, KML, COLLADA → CityGML → IFC

The development of the GeoBIM -- CityGML extension

In 2009, Léon [6] demonstrated the latest application domain extension (ADE), which can convert the building information model (BIM) in the format of the open standard IFC into a CityGML (van Berlo, 2009). Because CityGML represents building information in a high level of detail, in order to represent the rich semantic information of IFC in CityGML, the researchers extend CityGML schemas with extra objects and properties. However, there are only few IFC classes that can be transformed in to CityGML extensions and have real meanings in it, because IFC and CityGML describe buildings in different representations and aspects.

Unified Building Model

The Unified Building Model (UBM) [9][1] represents the first fully integrated framework of IFC and CityGML in which IFC can be traced to CityGML and vice versa. The reference ontology in this study is defined as an expressive ontology for IFC and CityGML semantic models, which is a superset model that is extended to contain all the features and objects from both the IFC and CityGML building models, with respect to all levels of detail, including inner and outer spatial structures. The integration approach is performed in two steps: a building model is first converted from the source model into a UBM, which is then converted, from the UBM into the target building model. The UBM is considered as a standard schema, which generates mappings from both data sources to the standard schema. This standard schema can easily be extended if there is a demand for transformation from a new schema to an IFC or CityGML.

The BIM and GIS are intended to serve two different domains and scales: the former focuses on inner building structural components, while the latter is used for mapping the surrounding real outer world, and, since they use different representations to describe the same spatial objects, they have very diverse objects, classes, and properties. In order to achieve an accurate and efficient integration, most of the previous works only focus on the main structural components, such as walls, roofs, doors, windows, and so on, of which IFC and CityGML have the same semantic and geometric representations. Even though some entities can be semantically mapped from IFC to CityGML, it is still difficult to create the geometric matching relations between them. For example, a beam that runs across two rooms is represented in CityGML as two thematic objects because it is observable from both rooms, but in IFC, it is modeled as one object. Because of the different geometric representations that IFC and CityGML use, sometimes researchers [25] apply an evaluation function on all the possible ambiguous conversions in order to come up with an optimal one. Even though the CityGML is capable of representing detailed building information in, at most, LoD4, and is extended to model noise, tunnels, bridges, hydro, and utility networks [26], it still cannot represent the mechanical elements. In addition, during the conversion process, the properties and parameters attached to the components need to be kept in the target schema, but CityGML is not capable of keeping them. Under the above restrictions, the complex schemas and components are beyond the scope of the current research, as well as any components with complicated geometric shapes.

Commercial software products

The Feature Manipulation Engine (FME) is “an application providing unrivalled format support for data translation and integration, and unlimited flexibility in data model transformation and distribution” [27]. It can read, analyze, translate, manipulate, and write spatial data in various geospatial data formats.

FME Desktop consists of the Universal Viewer, Data Inspector, Workbench, and Quick Translate components. While the Universal Viewer and Data Inspector can provide visualization of various data models, it cannot manipulate them. Universal Viewer can only handle 2D data and is the precursor to Data Inspector. Data Inspector, which can handle 2D and 3D geospatial data, was invented to replace Universal Viewer in the future, but it does not currently have all of the capabilities of Universal Viewer.

FME Quick Translate offers fast and simple translation between different data formats to facilitate the interoperability of spatial data, and the translation rules are all fixed without any customization. Unfortunately, its translation function is not sophisticated enough to achieve our goal.

FME Workbench is one of the most important and frustrating applications we have been using in our project. A typical workflow consists of deciding what data to read in (readers), how to translate the data (transformers), and what format to write out to (writers). The later section discusses the various transformations we have attempted to use to translate IFC data into the CityGML model, along with an evaluation of the results.

Besides being free and open source, the intention of the Building Information Model server (BIMserver) software [10] is to centralize information from any building related project. It uses IFC as its core standard building model and stores building information in the format of IFC in an underlying database, so it is possible to query, merge, and filter all the BIM models and generate IFC files on the fly. Because it also supports exporting functionality in various formats, including CityGML, we consider it as a conversion tool for converting from IFC to CityGML. The increasing research work that contributes to the integration of IFC and CityGML is driven by the demand for exchanging and sharing spatial information among different applications, some of which are focused on buildings, while others address a wide range of city areas. To summarize, the contributions of all the existing research on the integration of IFC and CityGML are: they all define semantic mappings between IFC and CityGML; [5][6][9] develop frameworks and algorithms to make an automatic bidirectional or unidirectional conversion, in terms of geometric and semantic information; the integration is done with regard to each level of details of CityGML[4] [5]. [6] also discusses the rich semantics of the IFC and how they can be applied to more detailed CityGML models.

Another commercial software product that converts from IFC to CityGML is IfcExplorer [7][28], which is an implementation of Nagel's conversion algorithm by the Research Centre Karlsruhe, at the Institute for Applied Computer Science in Germany. IfcExplorer is designed to make an automatic conversion of an IFC model into a CityGML, with regard to the selection of the specified LODs and building elements.

Chapter 4
Introducing our data integration system

In general, most common management, maintenance and inspection requests are related to equipment being out of order or in bad condition, or simply for a regular maintenance check. The basic information we need to perform the requests is about the equipment and its maintenance records. The equipment information consists of its location, its manufacture information, its manual, the utility network connecting to this equipment and so on, which is usually spread out from IFC to CityGML. In order to analyze the complex requests and solve them, we need to incorporate IFC and CityGML together. In our project, we have tried two approaches: transformation from one model to another or integrating both models together into a central data repository.

4.1 Transformation from IFC to CityGML

Our first approach was to evaluate and compare the currently available conversion tools and select the ones that are most suitable for our project, which is to reform the operation and maintenance system. After we analyzed all the converted results from the first step, we tried to complement the results by adding the missing parts to ensure that our system has sufficient information to answer all sorts of queries on facility management, maintenance operation, and inspection operation. Although this approach was not ultimately successful, it is instructive to see what we learned through this process.

After looking through the previous and ongoing conversion approaches (Chapter 3), we decided to use FME and BIMServer, because FME is a sophisticated software product and BIMServer is an open source program that can achieve our purpose. Other frameworks or applications that I mentioned above are either unavailable or not appropriate for our project.

FME Workbench provides various transformers to carry out all kinds of feature restructuring and data manipulating through format translation. It also supports the ability of users to change the schemas and mappings. Transformation occurs as the data is passed from reader to writer through a series of these transformers. We applied the geometric transformers on the building elements that we want to convert to CityGML, such as walls, roofs, windows, doors, to set the levels of details in CityGML, including their corresponding LoD names and defined feature types (e.g. walls, roofs…) that correspond to different LoDs. For instance LoD2 uses just walls and roofs, LoD3 adds doors and windows, LoD4 adds rooms. The conversion process is not always easy and accurate, because it is not a simple 1-1 correspondence between each building element. For example, in order to convert IFC space information into CityGML room information, we need to identify its associated walls, ceiling, and roof. It successfully creates the CityGML model in LoD3, but partly in LoD4, as viewed by some inspectors. We can see the buildings' facade textures on the exterior walls and roofs, as well as the stairs and interior walls, with doors and windows attached but they do not have room information, such as furniture, electricity units, and some complex geometric shaped building components (e.g. curtain walls).

"BIMServer is the ideal tool to support dynamic design, engineering, construction and building maintenance process." [10] Its core model server interprets all the building models in their IFC format, stores them in a common database, and exports them in various formats—CityGML being one option. We upload our IFC files to the BIMServer, and it exports the files in the CityGML format in LoD4 to us. By viewing them with the 3D model visualization tools, we found that the results do not conform to the standard CityGML schema. For example, in standard CityGML schemas, the floor surface is supposed to be part of the boundary surface, but in the result, the floor surface is attached to the room feature. Also, the number of floor surfaces and rooms cannot correctly match to the original IFC schemas. Besides, it does not contain the stairs, furniture, electricity units, and some complex geometric shaped building components as well.

The LandXplorer product developed by Autodesk is "a powerful software system for the management and visualization of geovirtual 3D city models and 3D landscape models" [29]. Not only does LandXplorer provide various functionalities that can explore, analyze, query, and navigate a 3D virtual city, but it also presents a fundamental rasterbased digital terrain model, on top of which it has additional geospatial data (buildings, plants, transportations) in various data formats like GIS and CAD, which are imported and integrated into the city model, as “city models”. LandXplorer can separately represent every LoD of the CityGML model by importing and applying each layer of texture and appearance gradually onto the basic LoD1 city model. LandXplorer offers an exploring panel, which provides a hierarchical view of the spatial objects and their attributes in the city model. In addition, it also supports spatial query functions to display desired buildings or spatial objects that meet certain criteria. For example, we can ask LandXplorer to display all the buildings built before a certain year or in a certain area. Furthermore, LandXplorer supports a connection to the 3D city database to import, export, and merge CityGML data.

Because of LandXplorer’s sophisticated navigation ability and its query function, our initial intension was to incorporate all related data models into CityGML and then import them into LandXplorer to execute complex queries and analyses.

We decided to use the IFC model of the CIRS building at UBC [30] as our experiment, which is recognized as a leader in sustainable buildings. It has completed documentations of both CAD and BIM files. After exporting these files to the IFC 2x2 format with AutoDesk Revit, the file size was too large to be uploaded to the BIMServer, so we cut off the architecture of the CIRS building into two floors, with the mechanical part. After converting it to the CityGML format with FME Workbench and BIMServer, overall, both CityGML files have part of the building components of Lod4, but they are not completed, and none of them can convert the mechanical part. I list the observations from the visualization of the resulted CityGML files:

- The mechanical part is totally missing in the CityGML model.
- The furniture part (chairs, desks) is also missing in the CityGML model.
- Stairs and railings are missing in the CityGML Model.
- The curtain exterior surface walls cannot be converted to CityGML components.
- The number of doors does not match.
- The properties of rooms (only in CityGML) do not match the space (only in IFC).

- For the CityGML model, while the walls, doors, and windows do have area measurement, they are really just geometric points coordinates, whereas in the IFC model, they are measured by width, length, and area.
- No storey information is in the CityGML Model.
- For each wall, window, door or room, there is not any associated information about storey level.

There were a number of reasons for this difficulty in what seems like it should have been a simple task. First, the BIM and GIS models are originally designed for different domains and purposes and to serve different areas of interest. Because of this, there is a significant technological barrier that prevents any automatic transformation between them. BIM is made for detailed building information, whereas GIS was designed to represent the real world in large scale, in an efficient and simple way. As a result, CityGML cannot accommodate much of the detailed information that IFC contains.

Most of the current research projects only consider the transformation of the building's architectural elements, such as walls, spaces, and doors, concentrating mainly on geometry transformation issues. To our knowledge, there is no systematized study on interoperability between the IFC and GIS systems for utility networks and mechanical elements.

The conceptual mappings between the IFC and CityGML components are too complicated to satisfy both a geometrical and semantic agreement. With regard to geometric representation, every single object in the BIM can be represented in two or even more different objects in CityGML with respect to different LoDs. Also, they use different geometric representations on the same object, which contributes to the difficulties of identifying the correspondence between them.

Because BIM is a new technology, older buildings were not designed using BIMs; many were created from CAD drawings. CAD models are not object oriented, nor do they carry rich semantic information about the spatial relationships among building elements, which makes the integration of building models into CityGML even harder.

In order to attempt to deal with some of the problems above, we turned to a CityGML extension, GeoBIM [23]. GeoBIM is the Application Domain Extension (ADE) for CityGML, achieved by adding features and properties from IFC, implemented by the BIMServer. We studied the GeoBIM project [6] and downloaded their code from Google Code. Their experimental results show that the extended objects (stairs) in the conversion results cannot be displayed in some inspectors, such as LandExplorer, but other viewers, such as FZK viewer, work well. After converting IFC to CityGML with the extension data, the size of the CityGML files is significantly larger than the original IFC files; in fact, they are increased by tenfold or more. Thus, CityGML cannot model heating and sewerage systems. Even if we could extend the CityGML schema to satisfy our needs and convert the data into CityGML, LandXplore cannot display them, because they are not included in the standard CityGML schemas. Other applications may not recognize them either. Therefore, we have come to the conclusion that forcing CityGML to accommodate whatever we want from IFC is not a good idea, since they are two different standard schemas which serve two different domains.

Even if we could add the BIM into the GIS—and there are some applications which can view them—the system would be very slow, because it would carry too much information. Nonetheless, the sum of the geospatial information on the whole campus and the detailed information about each building in the campus range is significantly large that it may exceed the maximum capacity of the current applications. Therefore, the efficiency of running the queries on them is very low, and it will affect the user experience. Hence, we think it is neither worthwhile nor necessary that we take all of this information into the system at the same time. At first, we need to access the geospatial information of the campus, in order to locate the specified building, and then we can load the detailed building information of that building and find the room that stores the problematic equipment. By thinking this way, we do not need to transfer BIM into GIS, and integrate two models together seems a more promising approach.

The Figure 5 below illustrates all the above approaches that we tried to transform building models into CityGML model and the comparison of the converted CityGML files from different tools.

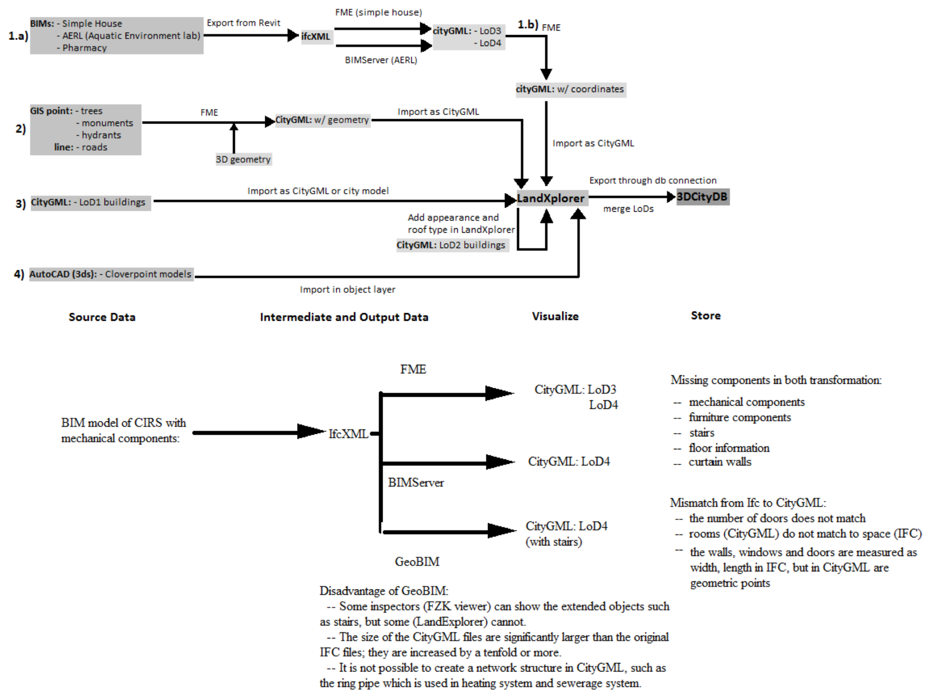

1.a)
BIMs: - Simple House
- AERL (Aquatic Environment lab)
- Pharmacy
Export from Revit
ifcXML
FME (simple house)
BIMServer (AERL)
cityGML: - LoD3
- LoD4
1.b) FME
cityGML: w/ coordinates
Import as CityGML
2)
GIS point: - trees
- monuments
- hydrants
line: - roads
FME
3D geometry
CityGML: w/ geometry
Import as CityGML
3) CityGML: - LoD1 buildings
Import as CityGML or city model
LandXplorer
Export through db connection
merge LoDs
3DCityDB
Add appearance and
roof type in LandXplorer
CityGML: LoD2 buildings
4) AutoCAD (3ds): - Cloverpoint models
Import in object layer
Source Data
Intermediate and Output Data
Visualize
Store
BIM model of CIRS with
mechanical components:
IfcXML
FME
CityGML: LoD3
LoD4
CityGML: LoD4
BIMServer
CityGML: LoD4
(with stairs)
GeoBIM
Missing components in both transformation:
-- mechanical components
-- furniture components
-- stairs
-- floor information
-- curtain walls
Mismatch from Ifc to CityGML:
-- the number of doors does not match
-- rooms (CityGML) do not match to space (IFC)
-- the walls, windows and doors are measured as width, length in IFC, but in CityGML are geometric points
Disadvantage of GeoBIM:
-- Some inspectors (FZK viewer) can show the extended objects such as stairs, but some (LandExplorer) cannot.
-- The size of the CityGML files are significantly larger than the original IFC files; they are increased by a tenfold or more.
-- It is not possible to create a network structure in CityGML, such as the ring pipe which is used in heating system and sewerage system.

Figure 5: A summary of our approaches to transform IFC to CityGML

Our alternative approach is to build a data warehouse or a data integration system, which transfers and loads the databases that store all the related information of BIMs and GIS into a central data repository. First, we need to inspect and collect all the desired databases from a wide range of fields, such as building information, facility information, maintenance requests, and a geospatial 3D city model. In the rest of this section, we examine and compare the data integration and data warehouse systems to decide which one should be used in our project.

4.2 Data integration and data warehouse

There are two popular ways to gather data together from different sources into one central system; one is loading all the separate data into a central data warehouse, and then we just treat the data warehouse as a traditional database to post queries on. The other way is to keep the separate data sources at their original places; we generate a mediated schema as a virtual central schema for users to post queries on, and then the queries are rewritten to each data source.

4.2.1 Data warehouse

A data warehouse [31] is a central data repository, where all the separate data sources are stored. The data from the original sources is extracted through their own schemas and transferred to the format of the schema of the data warehouse, and finally loaded into the warehouse (ETL). After all the ETL procedures, the data warehouse works as a traditional database system, in which all the queries are posted directly over the warehouse schemas and then the traditional query optimization techniques should be applied directly .

4.2.2 Data integration

Instead of loading all the data from separate sources into one central location, a data integration system creates a mediated schema based on all the source schemas, which represent all the information from all the data sources in the system, similar to the data warehouse schema, which eliminates the duplicate attributes and contains the unique attributes specified to each schema. All the source data are still stored where they originally were, so it saves space and transforming time in contrast to the data warehouse. Users query data from a unified mediated schema without knowing the source schemas and then each query over the mediated schema is then translated into queries over the source schemas. The strategy of query rewriting is to find all the sources capable of answering the queries and combine them together. Then we can use the traditional query optimization to execute them.

The data warehouse requests to store all the source data in a central data repository, which consumes significantly large space. Because right now we only use small part of the data from both IFC and CityGML schemas, we do not want to waste such space to store unimportant data and the query executing time of our data integration system from our experiences is acceptable. The schema of the data warehouse is well designed so that it is capable of accommodating data from all the isolated sources, and frequent modification of the schema should be avoided, because updating a data warehouse schema is very expensive. Right now, we are developing an integration system and our source schemas are adjusted by different operation requests and we try to merge more possible data sources. Also, our testing data is very limited and mappings between two IFC and CityGML schemas are still under test, therefore, at current stage, we cannot generate a stable data warehouse system. Data warehouse is more for globalized enterprises, which have very similar schemas, but our project tries to merge two building models which have diverse representations of the same objects. The data integration system is more flexible in that its mapping language is able to represent the implicit corresponding relations. Therefore, we use the data integration architecture to incorporate IFC and CityGML models.

Chapter 5
IFC schema

In order to build our data integration system, first we need the IFC and CityGML schemas, respectively. The IFC schema defines “a specification for sharing data throughout the project life-cycle, globally, [not only] across disciplines and across technical applications” [20], but also between different software applications in civil engineering. Since the IFC schema covers information at every stage of a building’s lifecycle, only a small part of the information is closely related to our project. Our goal is to clarify the scope of our project, analyze the IFC model, define a project-specific schema, and extract the corresponding attributes from the original IFC schema specifications.

An IFC schema not only defines the spatial structures of building elements, their properties, and the relationships between them, but also a building’s service elements, in terms of 3D geometric information (shape representation, location) and non-geometric properties (material or texture properties).

The IFC data can be exported into an XML format – ifcXML, which takes advantage of the XML format. Even though XML is well known for its rich, expressive, and well-designed format, in general, XML files are significantly large and only small parts of them are needed. Therefore, XML is designed to facilitate the work of extraction, transmission, and merging of partial models. We can locate elements by tags and discover their relationships from its hierarchical structure. In addition, since the XML format is widely used everywhere right now, it is a good idea to use this format to share and exchange information with others. There are many sophisticated tools to extract information from XML files, and, more importantly, we can use XQuery to query information from XML, as we can use SQL to query from relational database tables. In addition, there are several database applications and servers that support reading and writing the ifcXML schema directly into a relational database.

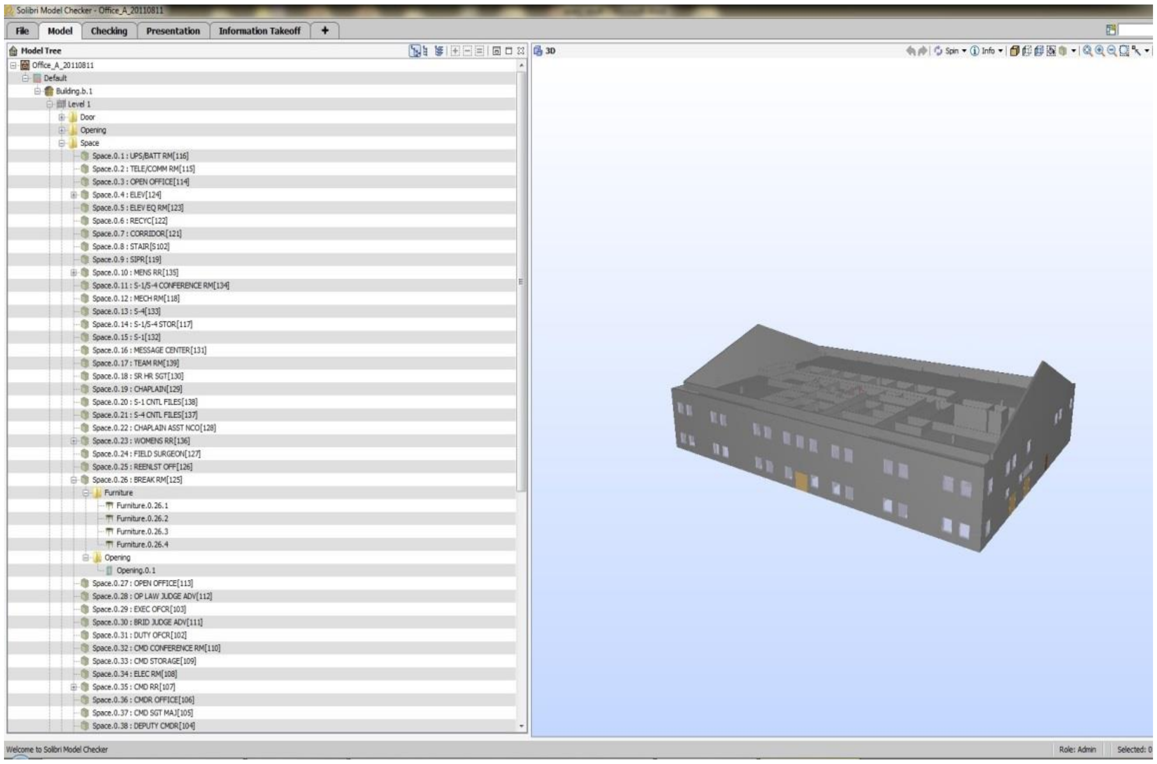
ifcXML's structure is much more complicated than some other IFC exporting formats; however, it contains the most information in terms of properties of elements, geometric relationships between elements, and covers project related information, and building service elements, etc. Since ifcXML files are significantly large and they have a complex format, the work of analyzing them is, of course, overwhelming. There is way too much information, and most of the data are not directly related to each other. Without domain experts' knowledge, it is impossible to extract a complete and accurate schema from it.

ifcXML represents the connections between objects or an object and its properties by reference IDs. Retrieving information is implemented by tracking the links of reference ID, and the reference paths can be very long. For example, in order to retrieve the information of a window on a given wall, we need to follow the reference path through IfcWallStandardCase, IfcRelVoidsElement, IfcOpeningElement, IfcRelFillsElement and IfcWindow by their individual IDs, as shown below.

However, even by tracing through the ID references, not all properties and relationships can be extracted from ifcXML. A large part of information is not explicitly expressed in ifcXML. For instance ifcXML cannot express directly if a wall is clipped or not, and to answer it, we need to analyze the shape representations of the walls. However, without experts' assistance, this is really beyond our scope. In this thesis, we only consider properties and relationships that can be derived by reference paths and simple arithmetic calculation. Since it is not necessary that we need to understand everything in ifcXML, we have decided to focus on the interesting information related to our project.

In reality, following the reference paths to extract the data based on the schema from ifcXML becomes really tedious and time-consuming. Therefore, we have to think about other options that are more straight-forward and efficient. We have since found out that Solibri Model Checker represents information exactly following IFC's standard specifications and provides all the information we want for our project.

Solibri Model Checker is a software application developed by Solibri, Inc., “the leader in Model Checking technology and innovation for design quality assurance and BIM analysis” [32]. As you can see below, it provides an easy-to-use visualization interface; the left panel displays all the elements in a building represented in a tree hierarchical structure and the right part demonstrates a complete 3D view of a building.
In our project, the IFC schema is generated from Solibri’s attribute sets with respect to each spatial object. There are some other similar tools, such as IFC Engine Viewer [33], a free IFC browser developed by TNO, Germany. However, Solibri Model Checker covers more information and provides more powerful visualization and analyzing functions.



```
Office_A_20110811
  Default
    Building.b.1
      Level 1
        Door
        Opening
        Space
          Space.0.1 : UPS/BATT RM[116]
          Space.0.2 : TELE/COMM RM[115]
          Space.0.3 : OPEN OFFICE[114]
          Space.0.4 : ELEV[124]
          Space.0.5 : ELEV EQ RM[123]
          Space.0.6 : RECYC[122]
          Space.0.7 : CORRIDOR[121]
          Space.0.8 : STAIR[S102]
          Space.0.9 : SIPR[119]
          Space.0.10 : MENS RR[135]
          Space.0.11 : S-1/S-4 CONFERENCE RM[134]
          Space.0.12 : MECH RM[118]
          Space.0.13 : S-4[133]
          Space.0.14 : S-1/S-4 STOR[117]
          Space.0.15 : S-1[132]
          Space.0.16 : MESSAGE CENTER[131]
          Space.0.17 : TEAM RM[139]
          Space.0.18 : SR HR SGT[130]
          Space.0.19 : CHAPLAIN[129]
          Space.0.20 : S-1 CNTL FILES[138]
          Space.0.21 : S-4 CNTL FILES[137]
          Space.0.22 : CHAPLAIN ASST NCO[128]
          Space.0.23 : WOMENS RR[136]
          Space.0.24 : FIELD SURGEON[127]
          Space.0.25 : REENLST OFF[126]
          Space.0.26 : BREAK RM[125]
            Furniture
              Furniture.0.26.1
              Furniture.0.26.2
              Furniture.0.26.3
              Furniture.0.26.4
            Opening
              Opening.0.1
          Space.0.27 : OPEN OFFICE[113]
          Space.0.28 : OP LAW JUDGE ADV[112]
          Space.0.29 : EXEC OFCR[103]
          Space.0.30 : BRID JUDGE ADV[111]
          Space.0.31 : DUTY OFCR[102]
          Space.0.32 : CMD CONFERENCE RM[110]
          Space.0.33 : CMD STORAGE[109]
          Space.0.34 : ELEC RM[108]
          Space.0.35 : CMD RR[107]
          Space.0.36 : CMDR OFFICE[106]
          Space.0.37 : CMD SGT MAJ[105]
          Space.0.38 : DEPUTY CMDR[104]
```

Figure 6: Solibri interface

Furthermore, we can easily define our IFC schema by simply look at their property sets, which not only contain all the related attributes of elements that can be found in ifcXML, both explicit and implicit, but also some implicit attributes derived from the original IFC schema and even some related elements referenced by the relationships.

After we define our IFC schema, there are two ways to extract the corresponding data; one is to parse the XML files by tags to extract the values from the given attributes, and the other way is to write the XQuery over the ifcXML files to get the data we need. Eventually, we will store the information into the relational database we build for the IFC data.

The first step in defining our IFC schema is to identify the scope of our project. In order to answer the queries regarding our case scenarios, we require the main spatial structure elements, such as building, building storey, space, wall…, and their material, geometric measurements, and some additional information about the equipment, such as equipment warranty information, for example.

In the IFC schema, the *IfcProject* is the uppermost container class and the highest spatial container class, such as the *IfcBuilding* or *IfcSite,* is immediately connected to the *IfcProject* by the *IfcRelAggregates* relationship. However, since the goal of this thesis is not to exhaustively describe IFC, we only describe *IFcBuilding*. This allows us see the general features and organization of IFC. The spatial containment hierarchy is shown in Figure 7, in which the *IfcBuilding* element consists of all the storeys in the building, and all the elements are aggregated according to their located storeys. The hierarchical structure of a building clearly represents the containment relationships between the building components on different levels.

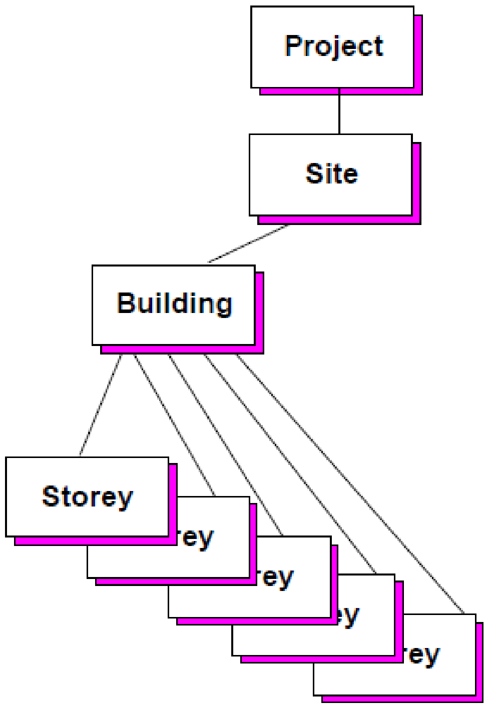


Figure 7: Spatial structure of a building project from [20]

While IFC represents a building as *IfcBuilding*, which provides building information, and *IfcBuilding* includes references to its building's storeys, *IfcBuildingStorey* is used to denote the containment relationship. *IfcBuildingStorey* also includes references to all the elements that belong to a storey. All the spatial and structural elements inherit the following basic attributes: name, type, objectPlacement, and representation. For *IfcBuilding*, in addition to basic information, there are four specific attributes: ElevationOfRefHeight "which is needed to put the elevation datum of the stories into the global context of the elevation above sea level" [20]. The ElevationOfRefHeight gives "the height relative to ± 0.00 (normally the elevation of the ground floor" [20]. The ElevationOfTerrain is the minimum elevation of the terrain on which the building stands and the BuildingAddress gives the actual address associated with this building.

I found a two-story office model in IFC developed by E. William East, a PhD of the Engineer Research and Development Center in the U.S Army Corps of Engineers [34]. Below is a part of the IFC model which has information on the basic spatial-structural elements, *IfcProject*, *IfcSite*, *IfcBuilding*, *IfcBuildingStorey*. As you can see, you cannot get much useful information from this–only the names of the building's storeys.

```
#49456=IFCPROJECT('27TOPmxCrDgPimmYFfwtvF',#1,'Project Number',$,$,'','Project Status',(#4,#9569),#51219);
#11652=IFCSITE('27TOPmxCrDgPimmYFfwtvD',#1,'Default',$,'',#9570,$,$,.ELEMENT.,(42,21,30,344238),(-71,-3,-35,-194702),-
0.0,$,$);
#3690=IFCBUILDING('27TOPmxCrDgPimmYFfwtvE',#1,$,$,$,#3678,$,$,.ELEMENT.,$,$,#17109);
#1116=IFCBUILDINGSTOREY('27TOPmxCrDgPimmYCM5828',#1,'Level 1',$,$,#28,$,$,.ELEMENT.,0.0);
#1117=IFCBUILDINGSTOREY('27TOPmxCrDgPimmYCM58C9',#1,'Level 2',$,$,#32,$,$,.ELEMENT.,4.267220999999476);
#1174=IFCBUILDINGSTOREY('27TOPmxCrDgPimmYCM5fAK',#1,'Roof',$,$,#1044,$,$,.ELEMENT.,7.924820999999477);
```

In order to discover the relationships between them, *IfcRelAggregates* is used to reveal the containment relations among *IfcProject* and *IfcSite*, *IfcSite* and *IfcBuilding*, *IfcBuilding* and *IfcBuildingStorey*.

```
#54554=IFCRELAGGREGATES('2VM08GTp95IBaI8JqySUbu',#1,$,$,#49456,(#11652));
#54553=IFCRELAGGREGATES('1LCR3ms7j60gBax59Dj$lD',#1,$,$,#11652,(#3690));
#62800=IFCRELAGGREGATES('2pTQXg5C14YAzX7Ek6JIH8',#1,$,$,#3690,(#1116,#1117,#1174));
```

Therefore, we define the building relation as:
IFC.Building (building_name, area, numStorey, fireRating, description, objectPlacement, elevationOfRefHeight, elevationOfTerrain, buildingAddress, projectIssueDate). The relation name and attribute names are meaningful for representing their semantic information. The containment relationship between the building and its storeys is expressed by a foreign key of "building ID" in the IFC Storey table.

Similarly, IFC describes the other components in the schema, including building components such as spaces and walls. All the relations in the IFC schema we design for our project are in the appendix.

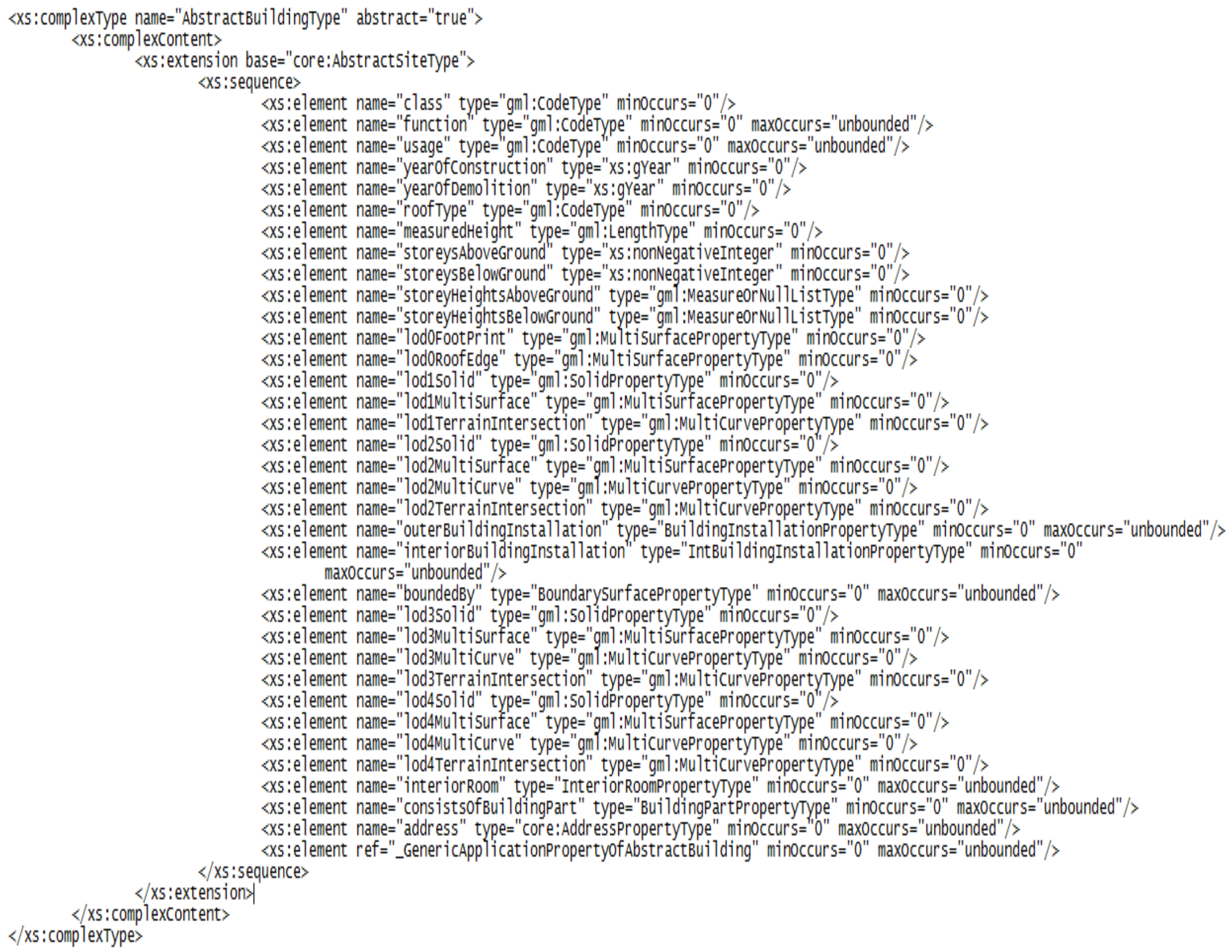
Chapter 6
CityGML schema

The CityGMLmodel is represented in an XML-based format to facilitate the exchange, share, storage, and maintenance of the virtual 3D city and landscape models. “It is an application schema for the Geography Markup Language version 3.1.1 (GML3), the extendible international standard for spatial data exchange issued by the Open Geospatial Consortium (OGC) and the ISO TC211” [35]. CityGML provides a comprehensive and extensive representation of the basic entities, attributes, and relations of a 3D city model in terms of their geometry, topology, semantics, and appearance. The same as in IFC, CityGML employs a hierarchical structure to generalize the relationships between spatial elements, as well as among aggregation and association relations. Unlike IFC, which is mainly focused on building information, CityGML is highly scalable and includes different urban entities from both a wide range and on different scales, such as whole sites, districts, cities, regions, and countries, in addition to individual buildings [36], but these are beyond the scope of our project. In order to incorporate the IFC schema, we only consider building related schemas.

Overall, CityGML represents spatial objects in a geometric model, an appearance model, and a thematic model. The geometric model defines the geometrical and topological representations of spatial objects in a 3D city model, and the appearance model describes the observable properties of the surface of an object in terms of material and texture. Furthermore, “the thematic model employs the geometry model for different thematic fields like Digital Terrain Models, sites (i.e. buildings, bridges, and tunnels), vegetation (solitary objects and also areal and volumetric biotopes), land use, water bodies, transportation facilities, and city furniture” [37]. CityGML is capable of being extended to express objects not defined in its standard specifications by either the concept of generic objects and attributes or Application Domain Extensions (ADE). The thematic model consists of many extension modules, which separately represent different thematic fields within a 3D city model: building, cityFurniture, bridge, generic CityObject, landUse, digital terrain model (relief), transportation, vegetation, and waterBody models.

As explained in the previous chapter, CityGML defines five different consecutive Levels of Detail (LoD), which describe different ranges of spatial objects with respect to the criteria of each LoD. Some objects can be represented simultaneously in more than one LoD, with different levels of geometric details according to the corresponding LODs. Furthermore, each object can have individual appearances corresponding to each LoD.

In CityGML, a "building inherits all the properties from AbstractBuilding: the building's class, function (e.g. residential, public, or industry), usage, year of construction, year of demolition, roof type, measured height, and the number and individual heights of all its storeys, above and below ground. Furthermore, *Addresses* can be assigned to Buildings or BuildingParts" [38].The figures below show the attributes of building in XML format.



```xml
<xs:complexType name="AbstractBuildingType" abstract="true">
        <xs:complexContent>
                <xs:extension base="core:AbstractSiteType">
                        <xs:sequence>
                                <xs:element name="class" type="gml:CodeType" minOccurs="0"/>
                                <xs:element name="function" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="usage" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="yearOfConstruction" type="xs:gYear" minOccurs="0"/>
                                <xs:element name="yearOfDemolition" type="xs:gYear" minOccurs="0"/>
                                <xs:element name="roofType" type="gml:CodeType" minOccurs="0"/>
                                <xs:element name="measuredHeight" type="gml:LengthType" minOccurs="0"/>
                                <xs:element name="storeysAboveGround" type="xs:nonNegativeInteger" minOccurs="0"/>
                                <xs:element name="storeysBelowGround" type="xs:nonNegativeInteger" minOccurs="0"/>
                                <xs:element name="storeyHeightsAboveGround" type="gml:MeasureOrNullListType" minOccurs="0"/>
                                <xs:element name="storeyHeightsBelowGround" type="gml:MeasureOrNullListType" minOccurs="0"/>
                                <xs:element name="lod0FootPrint" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
                                <xs:element name="lod0RoofEdge" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
                                <xs:element name="lod1Solid" type="gml:SolidPropertyType" minOccurs="0"/>
                                <xs:element name="lod1MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
                                <xs:element name="lod1TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="lod2Solid" type="gml:SolidPropertyType" minOccurs="0"/>
                                <xs:element name="lod2MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
                                <xs:element name="lod2MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="lod2TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="outerBuildingInstallation" type="BuildingInstallationPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="interiorBuildingInstallation" type="IntBuildingInstallationPropertyType" minOccurs="0"
                                        maxOccurs="unbounded"/>
                                <xs:element name="boundedBy" type="BoundarySurfacePropertyType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="lod3Solid" type="gml:SolidPropertyType" minOccurs="0"/>
                                <xs:element name="lod3MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
                                <xs:element name="lod3MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="lod3TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="lod4Solid" type="gml:SolidPropertyType" minOccurs="0"/>
                                <xs:element name="lod4MultiSurface" type="gml:MultiSurfacePropertyType" minOccurs="0"/>
                                <xs:element name="lod4MultiCurve" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="lod4TerrainIntersection" type="gml:MultiCurvePropertyType" minOccurs="0"/>
                                <xs:element name="interiorRoom" type="InteriorRoomPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="consistsOfBuildingPart" type="BuildingPartPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element name="address" type="core:AddressPropertyType" minOccurs="0" maxOccurs="unbounded"/>
                                <xs:element ref="_GenericApplicationPropertyOfAbstractBuilding" minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

Figure 8: Building attributes in the XML schema definition from [35].

Unlike in the IFC building schema, CityGML represents buildings and building components differently, with respect to each LoD. With the increasing LoDs, more geometric details and semantic objects are added. LoD1 only gives a generalized geometric representation of a building’s outer shell, but in LoD2 and LoD3, more geometric details and textures are added to the roofs and walls. These also include some exterior elements, like balconies and openings. LoD4 represents the interior of a building, the rooms, stairs, and furniture. Comparing with walls in IFC that are concrete volume objects, in CityGML walls are represented as surfaces, so they are distinguished as interior walls and exterior walls, and walls are categorized into many types of boundary surfaces by different functions, i.e., wall surface, roof surface, ground surface, floor surface, ceiling surface, interior wall surface and closure surface. Each surface has its corresponding geometric information.

3D City Database, which is based on a standard spatial relational database system (Oracle Spatial Database) is used to store, represent, and manage virtual 3D city models [39]. 3D City Database is designed to import, export and store spatial data in the format of CityGML by following the CityGML specification/schema. It supports five different levels of detail of CityGML representation and complex geometric and appearance data. 3D City Database plays an important role in our project as a database to store the CityGML model, and its schema is utilized in our data integration algorithms.

All the relations in the CityGML schema we design for our project are in the appendix.

Chapter 7
Schema mapping

Schema mapping is used in both data integration and data warehousing. The idea of schema mapping is to identify the same or similar attributes from all of the input schemas. The main criteria used in mapping are the similarity of names, data types, constraints, and schema structures, according to linguistic, structural and contextual identifiers [40]. While there are various datasets spread out all over the world today describing identical concepts, their schemas vary in both structure and name to a huge degree. Researchers have developed (and are still developing) schema mapping tools to define their similarities by identifying the corresponding attributes from all of them. The mapping can be one-to-one, one-to-many or indirect mappings by applying some operations on the attributes. Meanwhile, in order to guarantee the accuracy of the mappings, both names and contexts should be considered at the same time, because the same name can mean different things in different contexts. With the assistance of the thesaurus, linguistic mapping is able to identify short-forms (Qty for Quantity), acronyms (UoM for UnitOfMeasure), and synonyms (Bill and Invoice) [40]. Structural matching considers the similarities of the schema's contexts or inherent relations.

At the beginning, experts' knowledge made great contributions to the understanding of input schemas. There are many levels of mapping, such as schema or instance-based mapping, which consider only schema or instance data. Regarding schema based mapping, experts study the names, descriptions, domains, types, relationships, and constraints in the schemas. On the other hand, instance-based mapping statistically analyzes the overlaps of the input data to discover the related elements. Later on, in order to improve the accuracy of the mapping, user validation is needed. In addition, sometimes machine learning approaches are employed to train existing mappings to improve mapping performance.

There are many mapping tools available right now, such as Harmony [41], which is an open source schema matching tool that discovers the overlapping concepts and generates the correspondence across two data schemas. These kinds of matching tools are mainly used to identify the semantic correspondence by finding the similarity of the names of elements and attributes and the schemas' structure. Besides the semantic overlap, our two building schemas require domain experts' knowledge of understanding the spatial relationships between building components and their geometric properties. In the later section I explain that many researchers have identified the semantic overlapping concepts between IFC and CityGML already, so in our project, we will not use any mapping tools and we will focus on generating the correspondence regard to their geometric representations. Because of the two schemas involved Civil Engineering specific terminologies, different geometric and semantic representations, and complex relations between components, the schemas of building components are extremely complicated. First, however, we need to understand the building schemas well enough to distinguish the difference between IFC and CityGML representations, and then manually create the mappings.

After we define the IFC and CityGML schemas, respectively, the next step is to generate the schema mappings and the overlapping specifications, which identify the corresponding information from the two models. While both models describe the building information, they address different usages, different scopes, and have different semantic and geometric representations. Therefore, the overlapping specifications between the IFC and CityGML models are not simply one-to-one mappings, and most of them are indirect, in which some assistant functions and arithmetic expressions are applied. In previous research works, their mapping results between IFC and CityGML are mainly focused on the direct semantic mappings, but we are committed to developing implicit mappings on geometric representations, and later we modify existing algorithms [42] to use schema mappings to produce the mediated schema and mappings between mediated schema and source schemas. Finally, we implement the query rewriting algorithm to answer queries over the mediated schema, in order to achieve the goal of the integration of the IFC and CityGML models.

7.1 Schema mapping

Schema mapping defines a set of correspondences between two schemas, in terms of attributes and classes, in a declarative language. However, schema mapping in general is a difficult problem because the accuracy and efficiency of the existing approaches are hard to improve. Because schema mapping is a domain-specific problem, it benefits a great deal from the prior knowledge of the schema domain. However, in most cases, people cannot afford the huge expense of employing domain experts, so the current lack of understanding of the schemas leads to inaccurate and incomplete mapping results.

In our project, we could not find experts on both schemas, so we studied the schemas by reading their documentation. Since these schemas are in the Civil engineering domain, we cannot guarantee neither a comprehensive and precise understanding of the terminologies in this field, nor accurate expressions of overlapping information between two building schemas. Also, the rich and highly-detailed representations of buildings and their components increase the complexity of the schemas and the mapping specifications. Therefore, the schemas we define in our project are the part of the standard schemas that we guarantee that we understand and they are useful in our case scenarios, and the schema mappings we identify are based on the understanding of their respective meanings in each schema.

IFC and CityGML vary in their spatial-structural and geometric representations, that is, “IFC geometry uses constructive solid geometry with volumetric, parametric primitives representing the structural components of buildings, but CityGML uses boundary representations; accumulation of observable surfaces of topographic features” [6] (Figure 9: Geometry modeling paradigm differences between IFC (left) and CityGML (right) from). Every building element in CityGML is expressed by all the surfaces that construct them, but IFC is an element-based volume model, in that every building element is a concrete 3D spatial object. For example, a wall in IFC schema is a 3D concrete wall, but in CityGML a wall is represented as interior and exterior wall surfaces.

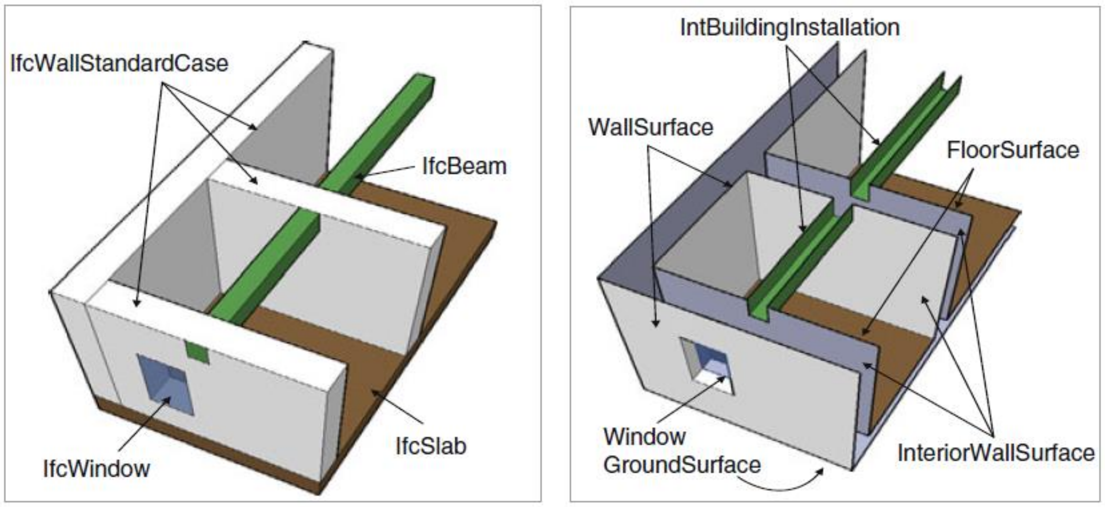


Figure 9: Geometry modeling paradigm differences between IFC (left) and CityGML (right) from [25]

As we can see, the IFC model uses space-enclosing structures, which means that it applies a hierarchy of structure with regard to their spatial enclosing relationships, in the order of building, storey, space, etc., yet CityGML defines all the building components in parallel under the building. Instead of using the concept of rooms, IFC, or, in this case, *IfcSpace,* defines an area or space as actually bounded by walls or an open area. Thus, the actual rooms in the building are not directly mapped to *IfcSpace* objects respectively. The IFC schema defines a building consisting of at least one storey, and all the spaces are attached to the storey they belong to. However, CityGML does not explicitly define the storey concept, but, instead, provides an aggregation function to group all building components on a certain height level into a storey class. CityGML defines a building installation concept to represent building objects of ramps, chimneys, balconies, beams, columns, etc, but, in IFC, they are expressed as individual building elements with different property sets.

In analyzing the above various differences, we have found it impossible to identify all the intersecting concepts in their properties and structures. Hence, we have limited the scope of our study by considering only spatial structural parts of buildings, omitting concepts such as project, site, and other construction-phase- related information. In our project, we focus only on the mappings, with regard to their semantic, conceptual, and geometric representations, but not their spatial structural relationships. Even though, we cannot cover all the possible overlapping information, the schema mappings we generate can still achieve our goal of facilitating decision making in building facility management and maintenance operations.

Referring to the previous works on the mappings of IFC and CityGML models, one approach to generate the mappings is between the classes of two models, with regard to their LoDs. As shown in Figure 10, in CityGML different LoDs define different coverage of the building components and details. Therefore, a set of mapping rules are defined with regard to each LoD. Meanwhile, the reference ontology defines the standard schema to guide the bidirectional transformation in [1].

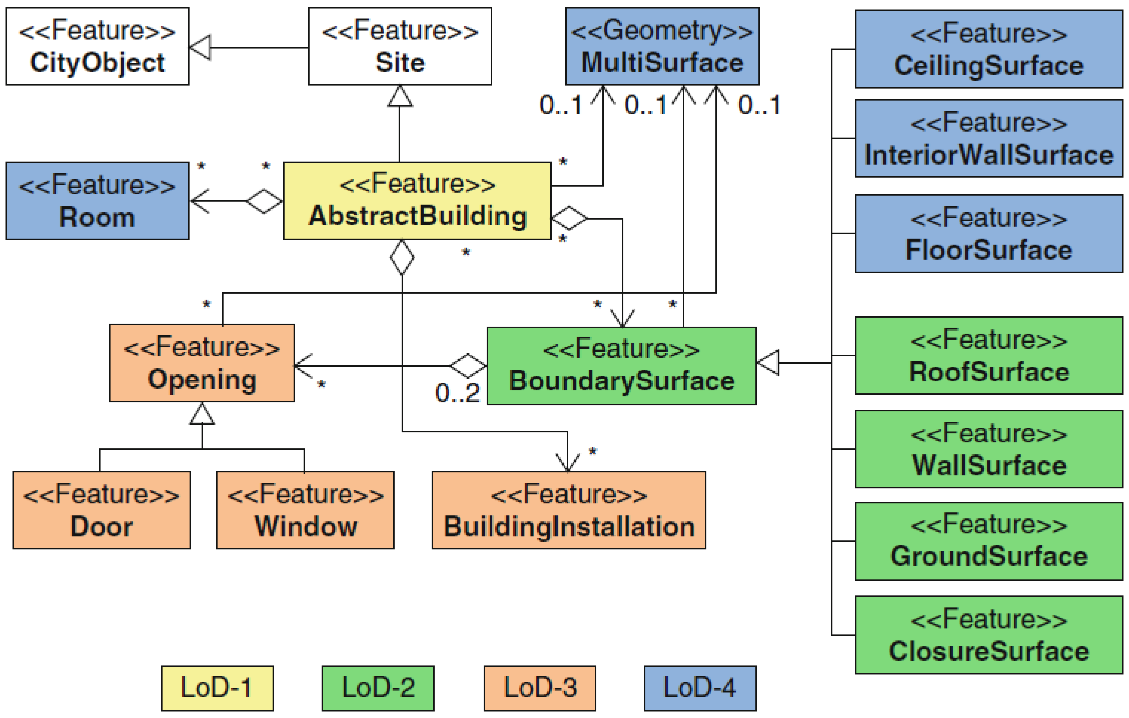
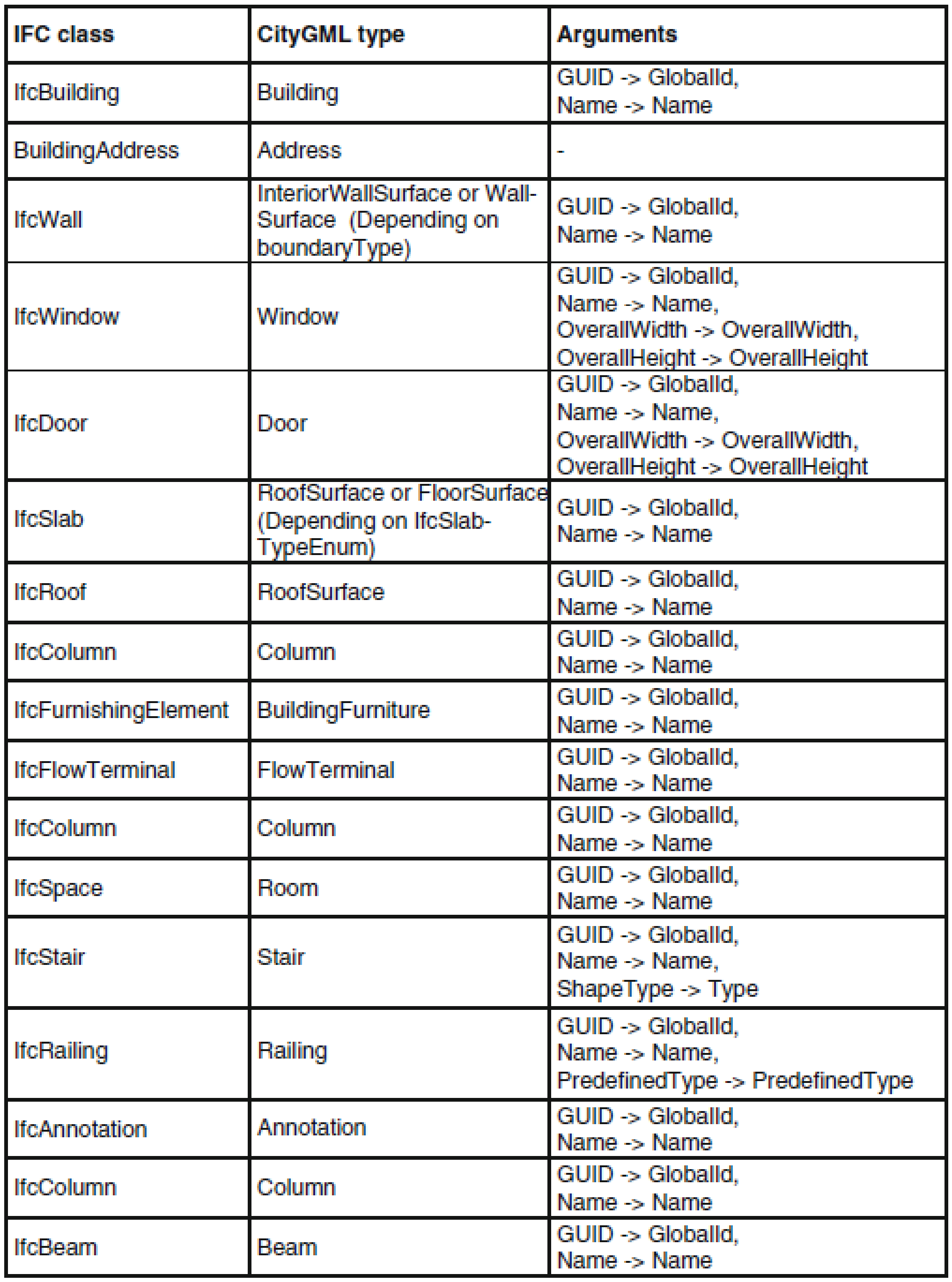

<<Feature>> CityObject
<<Feature>> Site
<<Geometry>> MultiSurface
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> Room
<<Feature>> AbstractBuilding
<<Feature>> Opening
<<Feature>> BoundarySurface
<<Feature>> RoofSurface
<<Feature>> WallSurface
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> Door
<<Feature>> Window
<<Feature>> BuildingInstallation
0..1
0..1
0..1
0..2
*
LoD-1
LoD-2
LoD-3
LoD-4

Figure 10: CityGML building model with regard to different LoDs from [1]

7.1.1 Existing mapping approaches for IFC and CityGML

In this section, we briefly discuss two existing approaches to mapping IFC and CityGML.

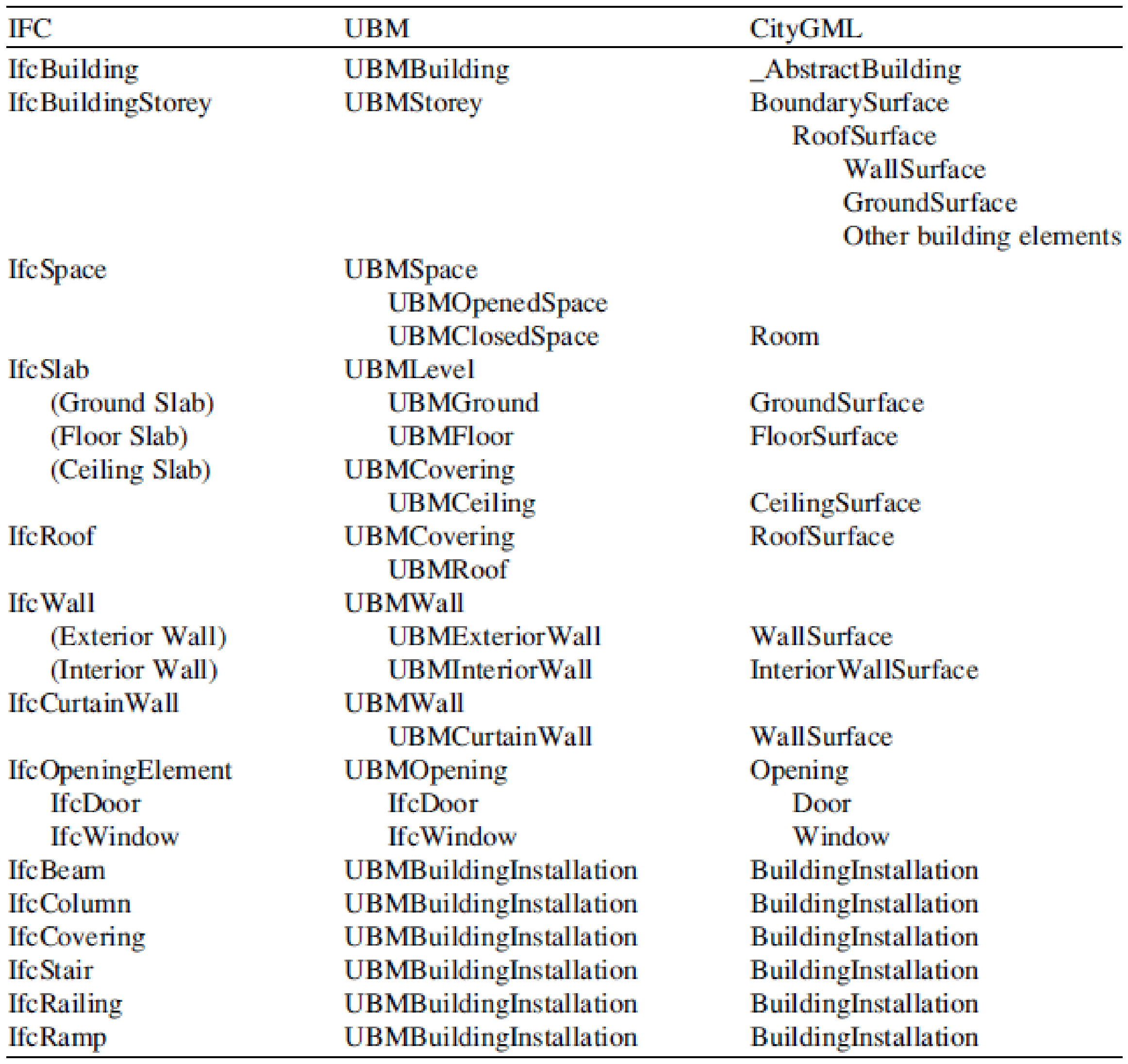
[6] generates the mappings of classes between IFC and CityGML models, including overlapping arguments and attributes (Table 1). It identifies the concepts from each model that represent the same objects, and their matching attributes only with regard to their semantic meanings. Nevertheless, this only produces direct mappings of concepts and their attributes, as in the result below, in which the mapping attributes are mostly just *GUID* and *name*. Additionally, the "CityGML type" column is not an actual concept in the CityGML schema, so the mapping is not on the schema level.



IFC class	CityGML type	Arguments
IfcBuilding	Building	GUID -> GlobalId, Name -> Name
BuildingAddress	Address	-
IfcWall	InteriorWallSurface or Wall-Surface (Depending on boundaryType)	GUID -> GlobalId, Name -> Name
IfcWindow	Window	GUID -> GlobalId, Name -> Name, OverallWidth -> OverallWidth, OverallHeight -> OverallHeight
IfcDoor	Door	GUID -> GlobalId, Name -> Name, OverallWidth -> OverallWidth, OverallHeight -> OverallHeight
IfcSlab	RoofSurface or FloorSurface (Depending on IfcSlab-TypeEnum)	GUID -> GlobalId, Name -> Name
IfcRoof	RoofSurface	GUID -> GlobalId, Name -> Name
IfcColumn	Column	GUID -> GlobalId, Name -> Name
IfcFurnishingElement	BuildingFurniture	GUID -> GlobalId, Name -> Name
IfcFlowTerminal	FlowTerminal	GUID -> GlobalId, Name -> Name
IfcColumn	Column	GUID -> GlobalId, Name -> Name
IfcSpace	Room	GUID -> GlobalId, Name -> Name
IfcStair	Stair	GUID -> GlobalId, Name -> Name, ShapeType -> Type
IfcRailing	Railing	GUID -> GlobalId, Name -> Name, PredefinedType -> PredefinedType
IfcAnnotation	Annotation	GUID -> GlobalId, Name -> Name
IfcColumn	Column	GUID -> GlobalId, Name -> Name
IfcBeam	Beam	GUID -> GlobalId, Name -> Name

Table 1: Mapping of IFC classes to CityGML types; including arguments and attributes from [6]

Another mapping approach [43] develops a new schema that falls between the IFC and CityGML schemas, as an assistant schema to generate building elements that overlap between the IFC and CityGML schemas (Table 2). In the table below, not only are most of the mapping elements the same as in Table 1, but it indicates some indirect overlapping elements, such as storey, stair, and so on. Instead of explicitly defining the storey and stair concepts, CityGML represents the corresponding concepts with an aggregation of a group of walls and building elements on the associated storey and building installment.



IFC	UBM	CityGML
IfcBuilding	UBMBuilding	_AbstractBuilding
IfcBuildingStorey	UBMStorey	BoundarySurface
		RoofSurface
		WallSurface
		GroundSurface
		Other building elements
IfcSpace	UBMSpace	
	UBMOpenedSpace	
	UBMClosedSpace	Room
IfcSlab	UBMLevel	
(Ground Slab)	UBMGround	GroundSurface
(Floor Slab)	UBMFloor	FloorSurface
(Ceiling Slab)	UBMCovering	
	UBMCeiling	CeilingSurface
IfcRoof	UBMCovering	RoofSurface
	UBMRoof	
IfcWall	UBMWall	
(Exterior Wall)	UBMExteriorWall	WallSurface
(Interior Wall)	UBMInteriorWall	InteriorWallSurface
IfcCurtainWall	UBMWall	
	UBMCurtainWall	WallSurface
IfcOpeningElement	UBMOpening	Opening
IfcDoor	IfcDoor	Door
IfcWindow	IfcWindow	Window
IfcBeam	UBMBuildingInstallation	BuildingInstallation
IfcColumn	UBMBuildingInstallation	BuildingInstallation
IfcCovering	UBMBuildingInstallation	BuildingInstallation
IfcStair	UBMBuildingInstallation	BuildingInstallation
IfcRailing	UBMBuildingInstallation	BuildingInstallation
IfcRamp	UBMBuildingInstallation	BuildingInstallation

Table 2: IFC – UBM – CityGML mapping from [43]

7.1.2 Our mapping approach

Having examined the existing approaches, we realized that they were not adequate for our needs because they only indicated the correspondence of the building components from both schemas, but they did not discover the overlapping information of their corresponding attributes. Additionally, they did not consider about elements inside the rooms, such as furniture or equipment, nor texture, material or geometric properties of the building components. Hence we chose a different approach. We first identify the mapping concepts from both schemas, which include the relations among building, building address, wall, door, stair, furniture, texture, and material. By comparing this with the mappings from previous works, we improve the schema mapping result by indicating the similarities in the texture and material properties of the building components in two models. While “texture” describes the outer appearance of building components, “material” can include an array of material layers, each of which states its material name and the thickness of its layer. Moreover, we generate the overlapping attributes for each concept and many of them are not simply direct mappings.

In addition to the directly corresponding relations in both schemas, we also discover some implicit shared or overlapping elements and their equivalent attributes. While CityGML does not define the individual relations associated with each type of wall surface, its THEMATIC_SURFACE table represents all the surfaces, in which the attribute TYPE indicates the specific boundary surface by its value ("ClosureSurface", "GroundSurface", "WallSurface", "RoofSurface", "FloorSurface", "InteriorWallSurface", "CeilingSurface"). Thus, while our project maps to all the corresponding tables in the IFC schema, we only list the IfcWall table, since the IfcSlab, IfcRoof, and IfcCurtainWall all have the same semantic and geometric representations as IfcWall in the IFC schema. Not only does the Building_Installation table in the CityGML schema describe the building components, such as balconies, chimneys, dormers, or outer stairs, but it also maps multiple tables in the IFC tables.

While CityGML does not explicitly represent the width and height of a building component, the IFC schema defines the geometric measurements associated with each building element. However, the geometric information of building components is stored in the table SURFACE_GEOMETRY in the CityGML schema, which has an attribute of GEOMETRY of type SDO_GEOMETRY type. SDO_GEOMETRY defines the geometry of any surface, which has a representation of polygons. As regards the Oracle spatial database documentation, the geometric measurements can be obtained from the respective functions:

The length or perimeter of a geometry	SDO_GEOM.SDO_LENGTH
The area of a two-dimensional polygon	SDO_GEOM.SDO_AREA
The maximum value for the specified ordinate (dimension) of the minimum bounding rectangle of a geometry object	SDO_GEOM.SDO_MAX_MBR_ORDINATE
The minimum value for the specified ordinate (dimension) of the minimum bounding rectangle of a geometry object	SDO_GEOM.SDO_MIN_MBR_ORDINATE
The volume of a three-dimensional solid geometry	SDO_GEOM.SDO_VOLUME

Table 3: geometric functions from the Oracle Spatial Database

In order to get the area of a polygon, a parameter called “Tolerance” is passed to the area function, which defines a level of precision with the spatial data, for example, SDO_GEOM.SDO_AREA(shape, 0.005), where the tolerance is 0.005, and the shape represents the polygon. Now, we build a connection between the two models in terms of geometric representation.

Taking the building concept as an example, we will explain how to identify mappings between two building relations. Also, the examples below explain how one attribute in one schema is equivalent to an expression of multiple different attributes in the other schema. The IFC and CityGML building relations are respectively defined as below:

IFC.BUILDING(BUILDING_NAME, AREA, NUMSTOREY, FIRERATING, DESCRIPTION, OBJECTPLACEMENT, ELEVATIONOFREFHEIGHT, ELEVATIONOFTERRAIN, BUILDINGADDRESS, PROJECTISSUEDATE)

CITYGML.BUILDING(ID, NAME, BUILDING_PARENT_ID, BUILDING_ROOT_ID, DESCRIPTION, CLASS, FUNCTION, USAGE, YEAR_OF_CONSTRUCTION, YEAR_OF_DEMOLITION, ROOF_TYPE, MEASURED_HEIGHT, STOREYS_ABOVE_GROUND, STOREYS_BELOW_GROUND, STOREY_HEIGHTS_ABOVE_GROUND, STOREY_HEIGHTS_BELOW_GROUND, LODX_TERRAIN_INTERSECTION, LODX_MULTI_CURVE, LODX_GEOMETRY_ID)

In CityGML building relations, the STOREYS_ABOVE_GROUND attribute refers to the number of storeys above the ground in this building, thus the NUMSTOREY in the IFC building relation is equal to:
STOREYS_ABOVE_GROUND+STOREYS_BELOW_GROUND to determine the total number of storeys in the building. The STOREY_HEIGHTS_ABOVE_GROUND attribute gives us an array of heights of storeys above the ground, ordered by their storey numbers. In addition, the information on roof type can be obtained from the TYPE attribute of IfcRoof relation, and the year of construction of the building is found in IfcWorkSchedule class.

In order to simplify the mappings, we only generate the mappings on the concepts of building, building address, wall, door, room, stair, furniture, texture, and material. We do not enumerate all the types of boundary elements, such as roof, ceiling, curtain wall and slab, and we use the wall concept to represent all of them because of their semantic and geometric similarities. We also omit the non-movable objects, such as interior stairs, railings and pipes, which are classified into the IntBuildingInstallation class in the CityGML schema. Moreover, in spite of specifying the schema mappings with regard to each different LoD, we generate one unified and comprehensive mapping on each concept.

In summary, our schema mappings make the following contributions to the integration of the IFC and CityGML models:

- Even though the IFC and CityGML have different geometric representations of their objects, we identify their basic corresponding measurements, for instance, length, height, width, perimeter, and area.
- In contrast to the previous mapping results, our mapping result enriches the overlapping information on attributes of the primary building components, by a thorough study of both schemas.
- We specify the material and texture information of building components in the mappings, in terms of texture type, texture image, texture coordinates, texture wrapping mode, transparency of material, and so on.
- We apply the arithmetical and complex mapping expressions to identify their implicit corresponding relationships.
- Since our schema mappings are bidirectional, we are later able to flexibly transform data between two models, starting from any schema first, and integrate them together into a comprehensive and rich building representation.

In the final analysis, however, there are many challenges remaining in our project that limit the quality and quantity of our mapping result. I will explain the difficulties that occurred in the mappings of the two building models in detail below. Most of the time, one building object is mapped to more than one object in the other schema. More often, multiple objects in the CityGML schema are mapped to a single IFC element [25]. Most likely, the one-to-many mappings happened between wall elements. There are two cases of one-to-many mappings between the IFC and CityGML schemas, as many IFC elements map to one CityGML element, and vice versa. While IFC is a component-based volume model, CityGML applies a boundary representation, which identifies an observable surface of topographic features as a single object. In Figure 11, the left wall surfaces are in the CityGML representation and the IFC walls are on the right. Wall surfaces, I21 and I22 are considered as two individual objects in CityGML, because they are observable from two rooms, which are represented as one solid wall in IFC. Another example is that the interior wall surfaces I11 and I13 are seen as two separated wall surfaces in CityGML because W1 is penetrated by another wall, W2, but combining them and the exterior surface of I12 together only reconstructs one concrete IFC wall object.

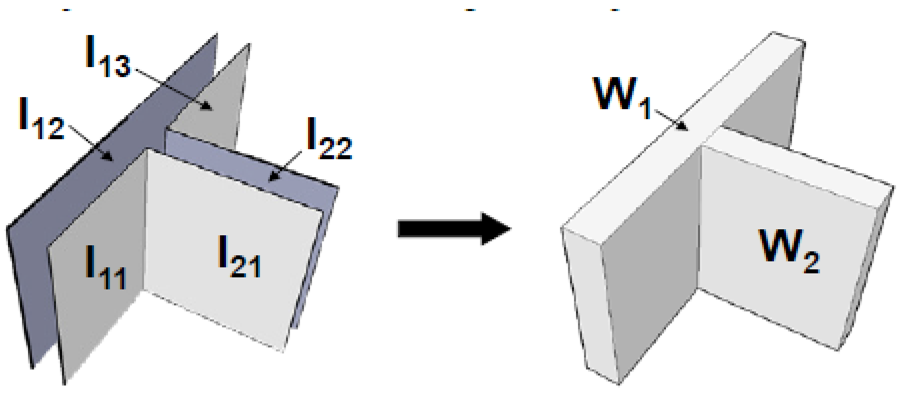


Figure 11: n: 1 match between CityGML wall surfaces and reconstructed IFC entities from **[25]**

In very few cases, one CityGML object is split into multiple objects in IFC. The building elements in IFC are grouped by their correspondingly located storeys, so the exterior fa çade wall of a building, spanning multiple storeys, is partitioned into one wall per storey. Other building components excluding walls are mostly one-to-one mapping, for instance windows, doors, furniture and so on.

In sum, in order to generate accurate mappings on the real building components between both models, on top of the semantic overlapping concepts, the geometric-topological relations define the geometric mapping criteria.

The schema-mapping language used in our project is a set of conjunctive queries in Datalog, which is a common approach in the schema mapping in database literature. Because Datalog is a really powerful language, it is capable of representing our complex arithmetic mapping expressions. The schema mapping defines any overlapping elements of the same concept in terms of source relations. The heads of a mapping on one concept are the same, which is a relation of all the overlapping attributes from all the relations on this concept, and the bodies of each query are expressions over relations on this concept from each respective schema.

Taking the mapping on building address as an example:

BuildingAddress(BUILDING_NAME, ADDRESSLINES, POSTALBOX, CITY, REGION, POSTALCODE, COUNTRY):-IFC.BuildingAddress(BUILDING_NAME, PURPOSE, DESCRIPTION, ADDRESSLINES, POSTALBOX, CITY, REGION, POSTALCODE, COUNTRY)
BuildingAddress(BUILDING_NAME, HOUSE_NUMBER+" "+STREET, POSTALBOX, CITY, REGION, POSTALCODE, COUNTRY):-CityGML.Address(BUILDING_NAME, STREET, HOUSE_NUMBER, POSTALBOX, POSTALCODE, CITY, REGION, COUNTRY, XAL_SOURCE, MULTIPOINT)

Most of the overlapping attributes are direct semantic mappings, and there is one implicit mapping from ADDRESSLINES to HOUSE_NUMBER+" "+STREET, which indicates that the address line consists of house number and street name.

7.2 Mediated schema

The mediated schema provides a unified representation of all related source data for users to query over without having to know all the source schemas. The mediated schema is designed to preserve all the information in all the source schemas, which covers both the overlapping elements and the source-specific elements. By definition, the source-specific elements are the elements that only occurred in one schema and have no direct or indirect corresponding element in the other schemas.

In Pottinger and Bernstein [42] (hereafter called PB)'s paper, they proposed an approach to generate the mediated schema from source schemas and overlapping specifications. All the overlapping elements should be in the mediated schema. The source specific elements are represented in the mediated schemas in two ways, depending on how closely they are related to the overlapping elements. One is to simply add them to the mediated schema and the other way is to add the relations which contain them.

Example 1: we are using two source schemas, UBC and UW from [44], as shown below, and they have corresponding relations on conference papers, projects and labs.

```
UBC.conf-paper(title, venue, year, pages)
UBC.univ_proj(univ, project)
UBC.proj_area(project, area)

UW.conf-paper(title, venue, year, url)
UW.project(univ, project)
UW.proj_lab(project, lab)
```

For the paper relation, UBC and UW share some attributes in common, such as title, venue and year. “Pages” and “url” are source-specific attributes, because they only appear in one data source, either UBC or UW, not both. The datalog representation of the overlapping specifications of “paper” is:

```
conf-paper(title, venue, year):-UBC.conf-paper(title, venue, year, pages)
conf-paper(title, venue, year):-UW.conf-paper(title, venue, year, url)
```

In the above example, the mediated schema should be extended to contain the overlapping attributes and source-specific attributes, because the source-specific attributes are also part of the properties of the paper schema. The relations’ names in the mediated schema start with the prefix “M.”; in this example the mediated schema is M.conf-paper(title,venue,year,pages, url), which is the union of the attributes from all the source schemas, which are duplicate free.

PB’s algorithm returns both a set of mediated schemas and a set of mappings that represent the corresponding relations between the mediated schema and the source schemas. The queries are posted only over mediated schema, and rewritten to become queries over source schemas. This algorithm guarantees that the returned query result can be from any single source data or a union of the results from both sources. Another paper [44] extends this work to merging of multiple schemas. Since we anticipate that future work with these schemas may require merging other sources, this is the algorithm and code we are extending. Unlike PB’s approach, the approach in [44]only returns overlapping concepts. Therefore, in order to query the result on one specific source, users need to ask queries over that source's schema.

PB's resulting mediated schema satisfies the following five correctness criteria [42]: completeness, overlap preservation, extended overlap preservation, normalization, and minimality, to ensure not only that the mediated schema represents all the information of the source schemas, but also that the queries over it can be rewritten to be answered by every single source or combination of all sources. Additionally, the mediated schema has the minimal set of relations of all the possible mediated schema sets that satisfy the first four criteria. In our project, our mediated schema algorithm is derived from [44], which is proved sound and complete for querying overlapping concepts. The algorithms from [44] are derived from PB's algorithms, and their algorithms still satisfy the above criteria because they preserve the core part of the algorithms which guarantee the five criteria. Our work is extending this by adding a set of arithmetic operations in the mappings, and in the following algorithms of generating the mediated schema, the correspondences between the mediated schema and the source schemas and the query rewriting, we adjust them to handle arithmetic operations instead of just attributes. Therefore, we did not change the basic ideas of the algorithms and our algorithm also satisfies the same properties of soundness and completeness.

Our algorithm for generating the mediated schema is the same as [44], so it has the same guarantees as their algorithm.

7.3 GLAV mappings

Two of the most influential mapping languages are global-as-view (GAV) and local-as-view (LAV), both of which represent the relationship between schemas as views. In GAV, the global sources are described as views on local source schemas. LAV defines local sources as views over global sources. Taking the paper schema as an example, the example below lists the GAV and LAV mappings over two source schemas of paper.

Example 2: while the GAV view defines the mappings between overlapping schema and source schemas, with regard to “paper” schema from [44], on the other hand, the LAV view describes the relations between source schemas and mediated schema.

```
GAV: conf-paper(title,venue,year,pages):- UBC.conf-paper(title,venue,year,pages)
GAV: conf-paper(title,venue,year,url):- UW.conf-paper(title,venue,year,url)

LAV: UBC.conf-paper(title,venue,year,pages):- M.conf-paper(title,venue,year,pages,url)
LAV: UW.conf-paper(title,venue,year,url):- M.conf-paper(title,venue,year,pages,url)
```

In the GAV mappings, the queries asked on the global schema are easy to rewrite to queries over sources schemas by simply replacing the global schema in the queries with the set of source schemas, as indicated in the GAV mappings. However, if we try to add a new source schema into the mappings, the overlapping schema has to be changed to be the new one; therefore, all the previous GAV views have to be changed to satisfy the new overlapping schema. The definitions for each mediated schema relation may need to be modified to describe how to retrieve data from the new sources.

Unlike GAV, it is easier to use LAV in the case of adding a new source by adding one LAV view describing the new source schema over the existing mediated schema. Nevertheless, answering queries is an NP-complete problem, since the local sources are described in terms of the mediated schema rather than the other way around. Translating the queries over mediated schemas needs to reverse the LAV mappings, which is discussed in many papers as a problem when answering queries using views.

In [42], the author proposes a limited global-local-as-view (GLAV) mapping between a mediated schema and the data sources, which combines the expressive power of GAV and LAV together for a completeness in relations and for extended overlap queries. Neither GAV nor LAV alone is adequate enough to express the relationships required between the mediated schema and the source schemas to rewrite the queries from over mediated schema to source schemas. In the following example, I will explain how to appropriately use GLAV mapping to rewrite queries, with regard to our complex overlapping specifications with arithmetic expressions. We express mappings over relational schemas as conjunctive queries, using Datalog notation. The conjunctive queries mean that each subgoal is connected to at least one subgoal, for example, subgoals e_i and e_j in query Q are connected when e_i and e_j have at least one variable in common, or e_i is connected to another subgoal, e_k, of Q that is connected to e_j.

PB's algorithm employs an intermediated schema as a help schema to connect between mediated schema and source schemas. The GLAV mappings from [42] are defined by the following rules:

GLAV Mapping $\text{Map}_{\underline{M\_S}}$:
LAV:
1. For each completeness relation M.R in M, Map_{M_S} includes the following query:
I.R(attr(M.R)) :- M.R(attr(M.R)).
2. For each relation I.R_i in I that corresponds to some overlap O in O, Map_{M_S} includes the following query:
$\text{I.R}_i(\text{attr}(\text{I.R}_i))$:- M.R(attr(M.R)) where M.R corresponds to O.
GAV:
1. For each completeness relation I.R in I, Map_{M_S} includes the following query:
R(attr(I.R)) :- R(attr(I.R)).
2. For each relation I.Ri in I that corresponds to some query Q_i in overlap O in O, Map_{M_S} includes the following query:
$\text{I.R}_i(\text{attr}(\text{I.R}_i))$:- body(Q).

Example 3: the GLAV mappings on the "paper" schema in [44] returned by PB's algorithm [42]

GLAV Mapping $\text{Map}_{\underline{M\_UBC}}$:
GAV: conf-paper(title,venue,year,url):- UBC.conf-paper(title,venue,year,url)
LAV: conf-paper(title,venue,year,url):- M.conf-paper(title,venue,year,url,pages)

GLAV Mapping $Map_{\underline{M\_UW}}$:

GAV: conf-paper(title,venue,year,pages):- UW.conf-paper(title,venue,year,pages)

LAV: conf-paper(title,venue,year,pages):- M.conf-paper(title,venue,year,url,pages)

In order to express the complete relationship between the mediated schema and source schemas, GLAV mappings consist of a pair of LAV and GAV for each source schema, which builds the connection between the mediated schema and each source schema with the assistance of an intermediated schema. The intermediate schema for each data source actually represents "the coverage of each mediated schema relation available from a particular source [45]". In our case, the mediated schema is the union of the attributes from all the source schemas on the same concept, so the intermediated schema for each source schema is the intersection of the mediated schema and the source schema, which means that the intermediated schema is really the source schema itself. In example 3, the intermediated schema in the Mapping MapM_UBC is the same as the UBC paper schema, so is UW paper schema. The body of the GAV and LAV mapping are composed, respectively, of source schema and mediated schema.

7.3.1 Our GLAV mapping approach

In our project, because of the arithmetic expression in the overlapping mappings, there is small modification made to the GLAV mapping in [42]. In example 4 below, the value of the "addressLines" attribute in the IFC BuildingAddress schema is equal to the expression, house_number+" "+street, both of which are in the CityGML BuildingAddress schema. The resulting mapping between the mediated schema and IFC source schema is the same as returning by PB's algorithm. However, the GLAV mapping referring to the mediated schema and CityGML source schema should be modified by adding "addressLines" to its intermediated schema. The mediated schema remains the same in the LAV view's body, but the equation "addressLines=house_number+" "+street" should be adhered to the end of the body of the GAV view, to ensure that the information of "addressLines" answered by the CityGML BuildingAddress schema as the expression of house_number+" "+street.

Example 4: the GLAV mappings on BuildingAddress schema

GLAV Mapping Map$_{\underline{\text{M\_IFC\_BuildingAddress}}}$:
GAV:
Q_IFC_BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country):-
IFC.BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country)
LAV:
Q_IFC_BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country):-
M.BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint)

GLAV Mapping Map$_{\underline{\text{M\_}}\text{CityGML\_ BuildingAddress}}$:
GAV:
Q_CityGML_BuildingAddress(building_name,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint,addressLines):-
CityGML.Address(building_name,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint), addressLines=house_number+" "+street
LAV:
Q_CityGML_BuildingAddress(building_name,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint,addressLines):-
M.BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint)

By using the arithmetic expressions, the implicit mappings are discovered from both schemas. Now, our GLAV mappings are able to return more information than the simple one-to-one attribute mappings. Because we learned that IFC and CityGML contain many of this kind of implicit mappings, by applying our modified GLAV mappings, we can reveal richer and more complete information even when information from one schema is missing, we can still use the other schema to return the desired information.

Our arithmetic expressions are in the GLAV mappings of every building component, also in the material and texture to discover the geometric properties of building components and their surface textures.

7.4 Query Rewriting

The rewriting process is straight-forward, and can be achieved by combining the GAV and LAV mappings together, as explained in our algorithm. Our algorithm is based on the main procedure from PB's query rewriting algorithm [42] and the modification we made is that we accept queries over the source schemas. There are two possible situations of posting a query, either over a source schema or over the mediated schema.

In PB's algorithm, the queries are only posted over the mediated schema, and their mediated schema can handle all cases. If you ask only about the overlapping attributes in the query, then it will return the union of results from all the source schemas. If you want to find information from one specific source, ask a query about the attributes specific only to that source schema, then select the attributes from the source you want. If at least one source schema does not have any source-specific attributes, then this source schema must be added to the mediated schema; otherwise, the algorithm cannot return information only from this source schema.

In our algorithm, if the queries are asked over a source schema, there is no rewriting needed in this case. Nonetheless, when queries are asked over the mediated schema, the rewriting strategy is to first look at all the LAV views which have this mediated schema in their body, and then find the corresponding GAV views, which have the same heads as the resulting LAV views' heads. The rewriting part is to replace the mediated schema with all the possible source schemas in the GAV views returned in the previous step. If the query is conjunctive, which has more than one subgoal in the body of the query, the bodies of the consequential rewriting queries are the result of all combinations of possible source schemas replacing each subgoal in the queries. Finally, the variable names in the source schemas are replaced if they are different from the corresponding ones in the queries. There is one important change in our algorithm. We must check the variables in all the subgoals of the rewritten queries to see if they contain all the variables in the query's head. If not, we delete that query. Because the query is posted over the mediated schema, which is the union of variables from all the source schemas, the source schemas used to replace it may not have the variables in the query's head, and we should not keep this source in our result set, because it cannot answer the query.

Example 5: a query q over the mediated schema of "BuildingAddress" by using the GLAV mappings from Example 4:

q(building_name, addressLines):- M.BuildingAddress(building_name,purpose, description, addressLines,postalBox,city,region,postalCode,country,street,house_number, xal_source,multipoint)
can be translated using the GLAV mappings Map$_{\underline{\text{M\_IFC\_BuildingAddress}}}$ and Map$_{\underline{\text{M\_}}\text{CityGML\_BuildingAddress}}$ into:
q(building_name, addressLines):- IFC.BuildingAddress(building_name,purpose, description,addressLines,postalBox,city,region,postalCode,country)
q(building_name, addressLines):- CityGML.Address(building_name,postalBox,city, region,postalCode,country,street,house_number,xal_source,multipoint),
addressLines=house_number+" "+street

In reference to the GLAV mappings in Example 4, in which there are two LAV views over the mediated schema, M, by looking at their corresponding GAV views, both source schemas can answer the "building_name" and "addressLines" information. Then we will rewrite the original query to two queries over the two source schemas, respectively. As shown in example 5, the arithmetic expression "house_number+" "+street" is calculated to answer attribute "addressLines" in the CityGML schema, which is not able be answered by any single attribute in the CityGML schema. The arithmetic expressions are easy and fast to calculate in the standard query language, and they do not increase the complexity of the query rewriting algorithm, because the body of the GAV view replaces the mediated schema no matter how many arithmetic expressions the body has.

The time complexity of the rewriting algorithm depends on the number of source schemas in the mappings and the number of subgoals in the query's body. The running time would include how long it takes to produce the number of source schemas in the mappings and the number of subgoals in the query body, which, in real life, would not be that long, as both numbers are not significantly large. Meanwhile, the experience from the result of PB's paper demonstrates the efficiency of query rewriting of our limited GLAV views in the general case. In addition, the materialized view can help to speed up the running time. Even though our query rewriting algorithm has proved to return the most certain answer, because our core idea is based on PB's, and it guarantees the reconstruction of conjunctive queries that are sound and complete, we cannot claim that our mediated schema is able to answer all queries. In sum, not only are our modified algorithms as efficient as the PB's algorithms [44], but also they significantly improve the capability of answering queries about building components from both IFC and CityGML schemas by applying arithmetic expressions to express the overlapping information.

Chapter 8
Summary

UBC's Operation Service Center receives all sorts of requests on facilities, including maintenance requests, management requests, and inspection requests. These facilities are widely located around the whole campus, and their connections are represented in the utility networks. These increasing complex operations and maintenance requests draw our attention to the demand for sophisticated and efficient integration frameworks and technology for sharing and exchanging information about the building and its outdoor geospatial environment. There are two major information models to describe the spatial objects around us, such as buildings, streets, lands, city furniture, and transportation: BIM and GIS. A BIM is all about building details; it is a semantic representation of buildings and their inner structural components. However, GIS is a 3D city model, which is a digital representation of the earth's surface and its spatial objects. Integrating building details into their broader contexts seems a very promising solution to fulfill the demand for applications that will help with the management and maintenance of the facilities and utilities.

Taking advantage of the previous integration approaches and the current 3D city visualization and management applications, we initially decided to build our system on top of the BIMServer or FME WorkBench, which seemed to be the top two of the most promising available applications. After getting the converted CityGML files from them, we looked through their content and hierarchy. In summary, their building models can achieve the details of LoD3, but only part of LoD4, as they do not have the inner furniture, stairs, and electricity units. Neither of them could convert the mechanical components in the buildings. In order to answer the queries about finding a mechanical system, we needed to add this part into the current CityGML files. First we thought of GeoBIM, which is an extension of CityGML. But because the standard CityGML schemas do not cover the mechanical and utility parts, we needed to design extended schemas first and then implement them, which took a lot of effort. In addition, the extended parts could not be successfully displayed in the current CityGML visualization applications, so we could not utilize them to analyze the operation and maintenance requests.

After comparing data integration systems and data warehouses, we decided to create our interoperable building model based on the data integration architecture. We defined the IFC and CityGML schemas from their standard schema specifications, respectively, and identified the overlapping information from them using Datalog notation. We applied complex arithmetic expressions, not only on semantic representations, but also on geometric representations of building components. After modifying PB's mediated schema generation and query rewriting algorithms, we generated the mediated schema and the corresponding relationships between the mediated schema and all the source schemas, respectively, with respect to mappings with arithmetic expressions. After running some queries on the query rewriting algorithm, we demonstrated that the queries posed over the mediated schema would be answered over all the sources that contain the information asked in the queries, and we prove the implicit information is able to be revealed from the source schema.

As there are still some challenges remaining in our project, we would like to propose some possible ideas for future research agendas.

In this project, we have considered integrating only two source schemas, but if there are more than two schemas to start with, or we try to dynamically add more data sources to collaborate with, our approach would not be capable to handle what we set out to do. Jie's paper [44] proposes a data integration architecture on the Peer Data Management System, MePSys, to efficiently and automatically maintain and update mediated schemas whenever changes are made to the system, such as new sources coming in, an existing source leaving, or source schemas being changed. Her system guarantees the efficiency and correctness of the mediated schema generation algorithm, so that no matter what orders the new sources are added to the system in, the final mediated schema will remain the same. Queries can be posted either over any peer schema in the system or over the mediated schema, and they will be executed by translating them, using the mappings over all the peers in the system to return a complete result. We can adjust our system based on her idea to accommodate more data sources and efficiently update the mediated schema and mappings when changes made to the source schemas.

A spatial query language can be used to identify the spatial structure relationships among building components. [58] proposes a feature-based framework to support user-driven queries on construction management functions. Many features and properties that cannot be explicitly retrieved from the IFC files are derived from analyzing the geometry and topological relationships between objects in the IFC model, such as identifying whether a wall is curved or clipped, or determining column or duct penetration. The query execution is implemented by performing it in XQuery – the standard XML query language on ifcXML files. A spatial query language is introduced to support "metric (closerThan, fartherThan, etc.), directional (above, below, northOf, etc.) and topological operators (touch, within, contain, etc.) for use in SQL statements" [47] along with their detailed implementations. We can try to use the spatial query language introduced in the above papers to derive more implicit geometric mappings that are not covered in our current results.

Rather than building data, utility data plays an important role in the maintenance and inspection operations. The utility networks, such as the pipe network, water network, steam network, sewerage network, are represented as GIS data by means of points and line segments. With the assistance of 2D topographical representations of utility networks, maintenance operators can determine the effects on other buildings of shutting down some equipment in a specific building by identifying the utility connections between different buildings. In our project, we ignored the utility data in our data integration system, because the standard schema specifications of both IFC and CityGML do not carry much utility related information. Both schemas provide some generic objects to store user-defined objects and properties. Therefore, one possible way to integrate utility data is to extend both schemas to accommodate utility data.

There is some other equipment-related information stored in several individual data repositories, such as manufacturer information, serial number, maintenance history, service manual, or spare part of the equipment that needs to be repaired or replaced. If we can incorporate all of them into our existing data integration system, we can enrich our mediated schemas so we can answer more maintenance related requests.

As I explain in section 8, we do not support many-to-many mappings between building objects, because this kind of mapping is hard to express on the schema level. Since we are now only considering mapping on the schema level, we need to extend our mapping to new instance levels in order to represent the many-to-many mappings on building components. Most likely, they need to be done manually, because the mapping rules will depend on various special cases.

Our mapping language is expressed in Datalog notation, but Datalog supports only the limited query operations of select, project, and join. By using another query language that can support more query operations, such as aggregation, for example, the schema mappings can express even more complicated corresponding relationships. Then, we would be able to acquire more implicit information from the source data.

If possible, with the assistance of domain experts' knowledge, schema mapping could be even more accurate and complete. If there are still some uncertain mappings, or more than one possible mapping on the same overlapping concept, a weighting function can be applied to evaluate the quality of the mapping results.

The above methods which improve the quality of the maintenance operation queries will also benefit maintenance operators by expediting their responses to them.

Bibliography

[1] M. El-Mekawy and A. Östman, "Semantic Mapping: an Ontology Engineering Method for Integrating Building Models in IFC and CITYGML," *Proceedings of the 3rd ISDE Digital Earth Summit*, pp. 12-14, 2010.
[2] Industry Foundation Classes, http://en.wikipedia.org/wiki/Industry_Foundation_Classes.
[3] Information of CityGML, http://www.citygmlwiki.org/index.php/Basic_Information.
[4] C. Nagel and T. H. Kolbe, "Conversion of IFC to CityGML," in *Meeting of the OGC 3DIM Working Group at OGC TC/PC Meeting, Paris (Frankreich)*, 2007, .
[5] U. Isikdag and S. Zlatanova, "Towards defining a framework for automatic generation of buildings in CityGML using building information models," in *3D Geo-Information Sciences*Anonymous Springer, 2009, pp. 79-96.
[6] R. de Laat and L. van Berlo, "Integration of BIM and GIS: The development of the CityGML GeoBIM extension," in *Advances in 3D Geo-Information Sciences*Anonymous Springer, 2011, pp. 211-225.
[7] IFCExplorer, http://www.ifcwiki.org/index.php/IfcExplorer_CityGML_Export.
[8] Safe Software from Wikipedia, http://en.wikipedia.org/wiki/Safe_Software.
[9] M. El-Mekawy, *Integrating BIM and GIS for 3D City Modelling: The Case of IFC and CityGML*, 2010.
[10] Building Information Model Server, http://bimserver.org/.
[11] FME Desktop documentation, http://docs.safe.com/fme/pdf/FMEGettingStarted.pdf.
[12] Data integration system, http://en.wikipedia.org/wiki/Data_integration.
[13] Building Information Modeling, http://en.wikipedia.org/wiki/Building_information_modeling.
[14] Geographic information system (GIS), http://en.wikipedia.org/wiki/Geographic_information_system.
[15] Building Information Modeling (BIM), http://www.buildingsmartalliance.org/index.php/nbims/faq/.

[16] Model - Industry Foundation Classes (IFC), http://www.buildingsmart.com/standards/ifc.
[17] NEXTMap Database, http://www.intermap.com/en-us/databases/nextmap.aspx.
[18] Candidate OpenGIS® CityGML Implementation Specification (City Geography Markup Language), https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/?artifact_id=16675.
[19] Slide from Léon van Berlo: CityGML Extension for BIM / IFC information, http://tools.bimtoolset.org/BIMserver/foss4g/20091022_FOSS4G-CityGML-IFC-ADE.pdf.
[20] L. Thomas, ifc 2x edition 3 model implementation guide, http://www.buildingsmart-tech.org/downloads/accompanying-documents/guidelines/IFC2x%20Model%20Implementation%20Guide%20V2-0b.pdf.
[21] Extensible Markup Language (XML), http://www.w3.org/XML/.
[22] Rachel A. Pottinger's slides: Data Integration: Querying Heterogeneous Information Sources Using Source Descriptions & Data Integration: The Teenage Years, http://www.cs.ubc.ca/~rap/teaching/534P/2011/slides/DataIntegration.pdf.
[23] I. Hijazi, M. Ehlers, S. Zlatanova, T. Becker and L. van Berlo, "Initial investigations for modeling interior utilities within 3D geo context: Transforming IFC-interior utility to CityGML/UtilityNetworkADE," in *Advances in 3D Geo-Information Sciences*Anonymous Springer, 2011, pp. 95-113.
[24] IFC for GIS from CityGML Wiki, http://www.citygmlwiki.org/index.php/IFC_for_GIS.
[25] C. Nagel, A. Stadler and T. H. Kolbe, "Conceptual requirements for the automatic reconstruction of building information models from uninterpreted 3D models," in *Academic Track of Geoweb 2009 Conference, Vancouver,* 2009, .
[26] CityGML Application Domain Extension from CityGML Wiki, http://www.citygmlwiki.org/index.php/CityGML-ADEs.
[27] Getting Started with FME® Desktop, http://docs.safe.com/fme/pdf/FMEGettingStarted.pdf.

[28] IFCExplorer from Karlsruhe Institute of Technology, http://www.iai.fzk.de/www-extern/index.php?id=1566.
[29] LandXplorer, http://www.3dgeo.de/landx.aspx.
[30] The Centre for Interactive Research on Sustainability (CIRS), http://cirs.ubc.ca/.
[31] Data warehouse, http://en.wikipedia.org/wiki/Data_warehouse.
[32] Solibri Model Checker, http://www.solibri.com/press-releases/solibri-introduces-solibri-model-checker-v8-1.html.
[33] TNO. IFC Engine Viewer, http://www.ifcbrowser.com/ifcengineviewer.html.
[34] Office Building IFC file - 2011-09-14-Office IFC 2x3 Coordination MVD, http://www.nibs.org/?page=bsa_commonbimfiles.
[35] OGC City Geography Markup Language (CityGML) Encoding Standard, http://www.citygml.org/.
[36] OGC City Geography Markup Language v 2.0, http://www.opengeospatial.org/pressroom/pressreleases/1599.
[37] Open Geospatial Consortium, Inc. Candidate OpenGIS® CityGML Implementation Specification (City Geography Markup Language), 2006-07-28, https://portal.opengeospatial.org/modules/admin/license_agreement.php?suppressHeaders=0&access_license_id=3&target=http://portal.opengeospatial.org/files/?artifact_id=16675.
[38] 3D-Geo-Database for CityGML, http://www.3dcitydb.net/?id=1897.
[39] 3D City Database, http://www.3dcitydb.net/index.php?id=1880.
[40] J. Madhavan, P. A. Bernstein and E. Rahm, "Generic schema matching with cupid," in *VLDB,* 2001, pp. 49-58.
[41] Harmony, http://openii.sourceforge.net/index.php?act=tools&page=harmony.
[42] R. Pottinger and P. A. Bernstein, "Schema merging and mapping creation for relational sources," in *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology,* 2008, pp. 73-84.
[43] M. El-Mekawy, A. Östman and K. Shahzad, "Towards interoperating cityGML and IFC building models: A unified model based approach," in *Advances in 3D Geo-Information Sciences*Anonymous Springer, 2011, pp. 73-93.

[44] J. Zhao, R. Pottinger, C. Brown and S. Rajagopalan, "Schema mediation in peer data management systems," *International Journal of Cooperative Information Systems,* vol. 20, pp. 261-305, 2011.

[45] R. A. Pottinger*, Processing Queries and Merging Schemas in Support of Data Integration,* 2004.

[46] A. Borrmann, "From GIS to BIM and back again–A spatial query language for 3D building models and 3D city models," in *5th International 3D Geoinfo Conference,*

Appendices

IFC schema:

IFC.Building(building_name, area, numStorey, fireRating, description, objectPlacement, elevationOfRefHeight, elevationOfTerrain, buildingAddress, projectIssueDate)

IFC.Roof(roof_name_above, number_above, type, building_name, roof_height_above, elevation_above, roof_color, categoryCode)

IFC.WorkSchedule(workSchedule_name, building_name, description, type, identifier, creationDate, creators, purpose, duration, totalFloat, year_of_construction, finishTime, workControlType, userDefinedControlType)

IFC.Storey(floor_name, floor_number, floor_height, floor_type, floor_elevation, building_name, categoryType, floor_color)

IFC.BuildingAddress(building_name, purpose, description, addressLines, postalBox, city, region, postalCode, country)

IFC.Wall(wall_name, description, building_name, floor, area, area_opening, height, length, thickness, volume, color, materialLayer, rooms, connectedWall, openingElements, isExternal, is_solid)

IFC.Door(id, name, description, door_type, building_name, room_name, operationType, floor, area, height, width, thickness, material, is_solid, isExternal, fireRating, color)

IFC.Space(space_name, number, description, building_name, room_type, floor, area, height, width, length, perimeter, volume, hasWindow, interiorOrExteriorSpace)

IFC.Stair(name, description, type, floor, building_name, bottomArea, numberRiser, numberTread, heightRiser, heightTread, isExternal, width, volume, material)

IFC.Furniture(name, description, material, floor, room_name, building_name, bottomArea, volume, perimeter, height, width, depth, color)

IFC.Texture(name, repeatS, repeatT, tex_texture_type, world_to_texture, width, height, tex_image_url, texture_coordinates)

IFC.Material(name, layerThickness, isVentilated, type, weight, thermalIrEmissivity, visibleTransmittance)

CityGML schema:

CityGML.Building(id, name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id)

CityGML.Surface_Geometry(lodX_geometry_id, gmlid, gmlid_codespace, parent_id, root_id, is_xlink, is_reverse, is_solid, is_composite, is_triangulated, geometry)

CityGML.Address(building_name, street, house_number, postalBox, postalCode, city, region, country, xal_source, multipoint)

CityGML.Thematic_Surface(wall_name, description, wall_type, building_id, room_id, lodX_multi_surface_id, opening_id)

CityGML.Room(room_id, room_name, description, class, function, usage, building_id, lodX_geometry_id)

CityGML.Opening(opening_id, opening_name, name_codespace, description, type, address_id, lodX_multi_surface_id)

CityGML.Opening_to_them_surface(opening_id, thematic_surface_id)

CityGML.Address_to_building(address_id, building_id)

CityGML.Building_Installation(id, isExternal, name, name_codespace, description, class, function, usage, building_id, room_id, lodX_geometry_id)

CityGML.Building_Furniture(name, name_codespace, description, class, function, usage, room_id, lodX_geometry_id, lodX_implicit_rep_id, lodX_implicit_ref_point, lodX_implicit_transformation)

CityGML.TextureParam(lodX_geometry_id, is_texture_parametrization, world_to_texture, texture_coordinates, surface_data_id)

CityGML.Surface_data(surface_data_id, gmlid, gmlid_codespace, surface_data_name, name_codespace, description, is_front, type, x3d_shinness, x3d_transparency, x3d_ambient_intensity, x3d_specular_color, x3d_diffuse_color, x3d_emissive_color, x3d_is_smooth, tex_image_url, tex_image, tex_mime_type, tex_texture_type, tex_wrap_mode, tex_border_color, gt_prefer_worldfile, ft_orientation, ft_reference_point)

Overlapping specifications between IFC and CityGML schemas:

Building(building_name, area, numStorey, description, year_of_construction, roof_type, sum(roof_height_above)+sum(roof_height_below), count(number_above), count(number_below), list(elevation_above), list(roof_height_below)):- IFC.Building(building_name, area, numStorey, fireRating, description, objectPlacement, elevationOfRefHeight, elevationOfTerrain, buildingAddress, projectIssueDate),IFC.Storey(floor_name_above, floor_number_above, floor_height_above, floor_type, floor_elevation_above, building_name, categoryType, floor_color),floor_elevation_above>=0,IFC.Storey(floor_name_below,

floor_number_below, floor_height_below, floor_type, floor_elevation_below, building_name, categoryType, floor_color),floor_elevation_below<0,IFC.WorkSchedule(workSchedule_name, building_name, description, type, identifier, creationDate, creators, purpose, duration, totalFloat, year_of_construction, finishTime, workControlType, userDefinedControlType),IFC.Roof(roof_name_below, number_below, roof_type, building_name, roof_height_below, elevation_below, roof_color, categoryCode)
<u>Building</u>(building_name, SDO_GEOM.SDO_AREA(geometry, 0.005), storeys_above_ground+storeys_below_ground, description, year_of_construction, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground):-CityGML.Building(id, name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id),CityGML.Surface_Geometry(lodX_geometry_id, gmlid, gmlid_codespace, parent_id, root_id, is_xlink, is_reverse, is_solid, is_composite, is_triangulated, geometry)

<u>BuildingAddress</u>(building_name, addressLines, postalBox, city, region, postalCode, country):-IFC.BuildingAddress(building_name, purpose, description, addressLines, postalBox, city, region, postalCode, country)
<u>BuildingAddress</u>(building_name, house_number+" "+street, postalBox, city, region, postalCode, country):-CityGML.Address(building_name, street, house_number, postalBox, postalCode, city, region, country, xal_source, multipoint)

<u>Wall</u>(wall_name, description, building_name, floor, area, perimeter, length, height, room_name, opening_name, is_solid):-IFC.Wall(wall_name, description, building_name, floor, area, perimeter, height, length, thickness, volume, color, materialLayer, rooms, connectedWall, openingElements, isExternal, is_solid)

Wall(wall_name, description, building_name, floor, SDO_GEOM.SDO_AREA(geometry, 0.005), SDO_GEOM.SDO_LENGTH(geometry, diminfo), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 1), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 2), room_name, opening_name, is_solid):-CityGML.Thematic_Surface(wall_name, description, wall_type, building_id, room_id, lodX_multi_surface_id, opening_id),CityGML.Building(building_id, building_name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id),CityGML.Room(room_id, room_name, description, class, function, usage, building_id, lodX_geometry_id), CityGML.Surface_Geometry(lodX_multi_surface_id, gmlid, gmlid_codespace, parent_id, root_id, is_xlink, is_reverse, is_solid, is_composite, is_triangulated, geometry),CityGML.Opening(opening_id, opening_name, name_codespace, description, type, address_id, lodX_multi_surface_id),User_sdo_geom_metadata(table_nmame, "geometry", diminfo, srid)

Door(name, description, building_name, room_name, area, perimeter, height, width, is_solid):-IFC.Door(id, name, description, door_type, building_name, room_name, operationType, floor, area, height, width, thickness, material, is_solid, isExternal, fireRating, color)

Door(name, description, building_name, room_name, SDO_GEOM.SDO_AREA(geometry, 0.005), SDO_GEOM.SDO_LENGTH(geometry, diminfo), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 1), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 2), is_solid):-CityGML.Opening(opening_id, name, name_codespace, description, "door", address_id, lodX_multi_surface_id),CityGML.Building(building_id, building_name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id),CityGML.Surface_Geometry(lodX_multi_surface_id, gmlid, gmlid_codespace, parent_id, root_id, is_xlink, is_reverse, is_solid, is_composite, is_triangulated, geometry),CityGML.Opening_to_them_surface(opening_id, thematic_surface_id),CityGML.Address_to_building(address_id, building_id),CityGML.Thematic_Surface(thematic_surface_id, name, description, type, building_id, room_id, lodX_multi_surface_id),CityGML.Room(room_id, room_name, description, class, function, usage, building_id, lodX_geometry_id),User_sdo_geom_metadata(table_nmame, "geometry", diminfo, srid)

<u>Room</u>(space_name+number, description, building_name, area, perimeter, length, width):-IFC.Space(space_name, number, description, building_name, room_type, floor, area, height, width, length, perimeter, volume, hasWindow, interiorOrExteriorSpace)
<u>Room</u>(room_name, description, building_name, SDO_GEOM.SDO_AREA(geometry, 0.005), SDO_GEOM.SDO_LENGTH(geometry), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry,DIMINFO,1), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 2)):-CityGML.Room(room_name, name_codespace, description, class, function, usage, building_id, lodX_geometry_id),CityGML.Building(building_id, building_name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id),CityGML.Surface_Geometry(lodX_multi_surface_id, gmlid, gmlid_codespace, parent_id, root_id, is_xlink, is_reverse, is_solid, is_composite, is_triangulated, geometry)

<u>Stair</u>(name, description, building_name, isExternal):-IFC.Stair(name, description, type, floor, building_name, bottomArea, numberRiser, numberTread, heightRiser, heightTread, isExternal, width, volume, material)

Stair(name, description, building_name, isExternal):-CityGML.Building_Installation(id, isExternal, name, name_codespace, description, class, function, usage, building_id, room_id, lodX_geometry_id),CityGML.Building(building_id, building_name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id)

Furniture(name, description, room_name, building_name, bottomArea, perimeter, height):-IFC.Furniture(name, description, material, floor, room_name, building_name, bottomArea, volume, perimeter, height, width, depth, color)
Furniture(name, description, room_name, building_name, SDO_GEOM.SDO_AREA(geometry, 0.005), SDO_GEOM.SDO_LENGTH(geometry), SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry,DIMINFO,1)):-CityGML.Building_Furniture(name, name_codespace, description, class, function, usage, room_id, lodX_geometry_id, lodX_implicit_rep_id, lodX_implicit_ref_point, lodX_implicit_transformation),CityGML.Room(room_id, room_name, description, class, function, usage, building_id, lodX_geometry_id),CityGML.Building(building_id, building_name, building_parent_id, building_root_id, description, class, function, usage, year_of_construction, year_of_demolition, roof_type, measured_height, storeys_above_ground, storeys_below_ground, storey_heights_above_ground, storey_heights_below_ground, lodX_terrain_intersection, lodX_multi_curve, lodX_geometry_id),CityGML.Surface_Geometry(lodX_geometry_id, gmlid, gmlid_codespace, parent_id, root_id, is_xlink, is_reverse, is_solid, is_composite, is_triangulated, geometry),CityGML.TextureParam(lodX_geometry_id, is_texture_parametrization, world_to_texture, texture_coordinates, surface_data_id),CityGML.Surface_data(surface_data_id, gmlid, gmlid_codespace, surface_data_name, name_codespace, description, is_front, type, x3d_shinness, x3d_transparency, x3d_ambient_intensity, x3d_specular_color, x3d_diffuse_color, x3d_emissive_color, x3d_is_smooth, tex_image_url, tex_image, tex_mime_type,

tex_texture_type, tex_wrap_mode, tex_border_color, gt_prefer_worldfile, ft_orientation, ft_reference_point),User_sdo_geom_metadata(table_nmame, "geometry", diminfo, srid)

<u>Texture</u>(name, tex_texture_type, tex_image_url, texture_coordinates, world_to_texture, wrap_mode(repeatS, repeatT)):-IFC.Texture(name, repeatS, repeatT, tex_texture_type, world_to_texture, width, height, tex_image_url, texture_coordinates)
<u>Texture</u>(name, tex_texture_type, tex_image_url, texture_coordinates, world_to_texture, tex_wrap_mode):-CityGML.Surface_data(surface_data_id, gmlid, gmlid_codespace, name, name_codespace, description, is_front, type, x3d_shinness, x3d_transparency, x3d_ambient_intensity, x3d_specular_color, x3d_diffuse_color, x3d_emissive_color, x3d_is_smooth, tex_image_url, tex_image, tex_mime_type, tex_texture_type, tex_wrap_mode, tex_border_color, gt_prefer_worldfile, ft_orientation, ft_reference_point),CityGML.TextureParam(surface_geometry_id, is_texture_parametrization, world_to_texture, texture_coordinates, surface_data_id)

<u>Material</u>(name, type, emissive(thermalIrEmissivity), transparency(visibleTransmittance)):-IFC.Material(name, layerThickness, isVentilated, type, weight, thermalIrEmissivity, visibleTransmittance)
<u>Material</u>(name, type, x3d_emissive_color, x3d_transparency):-CityGML.Surface_data(id, gmlid, gmlid_codespace, name, name_codespace, description, is_front, type, x3d_shinness, x3d_transparency, x3d_ambient_intensity, x3d_specular_color, x3d_diffuse_color, x3d_emissive_color, x3d_is_smooth, tex_image_url, tex_image, tex_mime_type, tex_texture_type, tex_wrap_mode, tex_border_color, gt_prefer_worldfile, ft_orientation, ft_reference_point)

Mediated schema:

<u>Building</u>(building_name,area,numStorey,fireRating,description,objectPlacement,elevationOfRefHeight,elevationOfTerrain,buildingAddress,projectIssueDate,floor_name_above,floor_number_above,floor_height_above,floor_type,floor_elevation_above,categoryType,floor_color,floor_name_below,floor_number_below,floor_height_below,floor_elevation_below,workSchedule_name,type,identifier,creationDate,creators,purpose,duration,totalFl

oat,year_of_construction,finishTime,workControlType,userDefinedControlType,roof_name_below,number_below,roof_type,roof_height_below,elevation_below,roof_color,categoryCode,id,name,building_parent_id,building_root_id,class,function,usage,year_of_demolition,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry)

Wall(wall_name,description,building_name,floor,area,perimeter,height,length,thickness,volume,color,materialLayer,rooms,connectedWall,openingElements,isExternal,is_solid,wall_type,building_id,room_id,lodX_multi_surface_id,opening_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,room_name,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,opening_name,name_codespace,type,address_id,table_nmame,"geometry",diminfo,srid)

Door(id,name,description,door_type,building_name,room_name,operationType,floor,area,height,width,thickness,material,is_solid,isExternal,fireRating,color,opening_id,name_codespace,"door",address_id,lodX_multi_surface_id,building_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,thematic_surface_id,type,room_id,table_nmame,"geometry",diminfo,srid)

Room(space_name,number,description,building_name,room_type,floor,area,height,width,length,perimeter,volume,hasWindow,interiorOrExteriorSpace,room_name,name_codespace,class,function,usage,building_id,lodX_geometry_id,building_parent_id,building_root _id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_multi_surface_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry)

<u>Stair</u>(name,description,type,floor,building_name,bottomArea,numberRiser,numberTread,heightRiser,heightTread,isExternal,width,volume,material,id,name_codespace,class,function,usage,building_id,room_id,lodX_geometry_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve)

<u>Furniture</u>(name,description,material,floor,room_name,building_name,bottomArea,volume,perimeter,height,width,depth,color,name_codespace,class,function,usage,room_id,lodX_geometry_id,lodX_implicit_rep_id,lodX_implicit_ref_point,lodX_implicit_transformation,building_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,is_texture_parametrization,world_to_texture,texture_coordinates,surface_data_id,surface_data_name,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,table_nmame,"geometry",diminfo,srid)

<u>Texture</u>(name,repeatS,repeatT,tex_texture_type,world_to_texture,width,height,tex_image_url,texture_coordinates,surface_data_id,gmlid,gmlid_codespace,name_codespace,description,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image,tex_mime_type,t

ex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,surface_geometry_id,is_texture_parametrization)

Material(name,layerThickness,isVentilated,type,weight,thermalIrEmissivity,visibleTransmittance,id,gmlid,gmlid_codespace,name_codespace,description,is_front,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point)

GLAV Mappings:

GLAV for IFC:

GAV:

Q_IFC_Building(building_name,area,numStorey,fireRating,description,objectPlacement,elevationOfRefHeight,elevationOfTerrain,buildingAddress,projectIssueDate,floor_name_above,floor_number_above,floor_height_above,floor_type,floor_elevation_above,categoryType,floor_color,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground):-
M.Building(building_name,area,numStorey,fireRating,description,objectPlacement,elevationOfRefHeight,elevationOfTerrain,buildingAddress,projectIssueDate,floor_name_above,floor_number_above,floor_height_above,floor_type,floor_elevation_above,categoryType,floor_color,floor_name_below,floor_number_below,floor_height_below,floor_elevation_below,workSchedule_name,type,identifier,creationDate,creators,purpose,duration,totalFloat,year_of_construction,finishTime,workControlType,userDefinedControlType,roof_name_below,number_below,roof_type,roof_height_below,elevation_below,roof_color,categoryCode,id,name,building_parent_id,building_root_id,class,function,usage,year_of_demolition,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry)

```
Q_IFC_BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country):-
M.BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint)

Q_IFC_Wall(wall_name,description,building_name,floor,area,perimeter,height,length,thickness,volume,color,materialLayer,rooms,connectedWall,openingElements,isExternal,is_solid):-
M.Wall(wall_name,description,building_name,floor,area,perimeter,height,length,thickness,volume,color,materialLayer,rooms,connectedWall,openingElements,isExternal,is_solid,wall_type,building_id,room_id,lodX_multi_surface_id,opening_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,room_name,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,opening_name,name_codespace,type,address_id,table_nmame,"geometry",diminfo,srid)

Q_IFC_Door(id,name,description,door_type,building_name,room_name,operationType,floor,area,height,width,thickness,material,is_solid,isExternal,fireRating,color):-
M.Door(id,name,description,door_type,building_name,room_name,operationType,floor,area,height,width,thickness,material,is_solid,isExternal,fireRating,color,opening_id,name_codespace,"door",address_id,lodX_multi_surface_id,building_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,thematic_surface_id,type,room_id,table_nmame,"geometry",diminfo,srid)
```

Q_IFC_Room(space_name,number,description,building_name,room_type,floor,area,height,width,length,perimeter,volume,hasWindow,interiorOrExteriorSpace,room_name):-
M.Room(space_name,number,description,building_name,room_type,floor,area,height,width,length,perimeter,volume,hasWindow,interiorOrExteriorSpace,room_name,name_codespace,class,function,usage,building_id,lodX_geometry_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_multi_surface_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry)

Q_IFC_Stair(name,description,type,floor,building_name,bottomArea,numberRiser,numberTread,heightRiser,heightTread,isExternal,width,volume,material):-
M.Stair(name,description,type,floor,building_name,bottomArea,numberRiser,numberTread,heightRiser,heightTread,isExternal,width,volume,material,id,name_codespace,class,function,usage,building_id,room_id,lodX_geometry_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve)

Q_IFC_Furniture(name,description,material,floor,room_name,building_name,bottomArea,volume,perimeter,height,width,depth,color):-
M.Furniture(name,description,material,floor,room_name,building_name,bottomArea,volume,perimeter,height,width,depth,color,name_codespace,class,function,usage,room_id,lodX_geometry_id,lodX_implicit_rep_id,lodX_implicit_ref_point,lodX_implicit_transformation,building_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,is_texture_parametrization,world_to_texture,texture_coordinates,surface_data_id,surface_data_name,is_front,type,x3d_shinness,x3d_transparency,

x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,table_nmame,"geometry",diminfo,srid)

Q_IFC_Texture(name,repeatS,repeatT,tex_texture_type,world_to_texture,width,height,tex_image_url,texture_coordinates,tex_wrap_mode):-
M.Texture(name,repeatS,repeatT,tex_texture_type,world_to_texture,width,height,tex_image_url,texture_coordinates,surface_data_id,gmlid,gmlid_codespace,name_codespace,description,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image,tex_mime_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,surface_geometry_id,is_texture_parametrization)

Q_IFC_Material(name,layerThickness,isVentilated,type,weight,thermalIrEmissivity,visibleTransmittance,x3d_emissive_color,x3d_transparency):-
M.Material(name,layerThickness,isVentilated,type,weight,thermalIrEmissivity,visibleTransmittance,id,gmlid,gmlid_codespace,name_codespace,description,is_front,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point)

LAV:
Q_IFC_Building(building_name,area,numStorey,fireRating,description,objectPlacement,elevationOfRefHeight,elevationOfTerrain,buildingAddress,projectIssueDate,floor_name_above,floor_number_above,floor_height_above,floor_type,floor_elevation_above,categoryType,floor_color,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground):-
IFC.Building(building_name,area,numStorey,fireRating,description,objectPlacement,elevationOfRefHeight,elevationOfTerrain,buildingAddress,projectIssueDate),IFC.Storey(building_name,floor_name_above,floor_number_above,floor_height_above,floor_type,floor

```
_elevation_above,categoryType,floor_color),measured_height=sum(roof_height_above)+
sum(roof_height_below),storeys_above_ground=count(number_above),storeys_below_gr
ound=count(number_below),storey_heights_above_ground=list(elevation_above),storey_
heights_below_ground=list(roof_height_below)

Q_IFC_BuildingAddress(building_name,purpose,description,addressLines,postalBox,cit
y,region,postalCode,country):-
IFC.BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,re
gion,postalCode,country)

Q_IFC_Wall(wall_name,description,building_name,floor,area,perimeter,height,length,thi
ckness,volume,color,materialLayer,rooms,connectedWall,openingElements,isExternal,is_
solid):-
IFC.Wall(wall_name,description,building_name,floor,area,perimeter,height,length,thickn
ess,volume,color,materialLayer,rooms,connectedWall,openingElements,isExternal,is_soli
d)

Q_IFC_Door(id,name,description,door_type,building_name,room_name,operationType,f
loor,area,height,width,thickness,material,is_solid,isExternal,fireRating,color):-
IFC.Door(id,name,description,door_type,building_name,room_name,operationType,floor
,area,height,width,thickness,material,is_solid,isExternal,fireRating,color)

Q_IFC_Room(space_name,number,description,building_name,room_type,floor,area,heig
ht,width,length,perimeter,volume,hasWindow,interiorOrExteriorSpace,room_name):-
IFC.Space(space_name,number,description,building_name,room_type,floor,area,height,
width,length,perimeter,volume,hasWindow,interiorOrExteriorSpace),room_name=space_
name+number

Q_IFC_Stair(name,description,type,floor,building_name,bottomArea,numberRiser,numb
erTread,heightRiser,heightTread,isExternal,width,volume,material):-
```

IFC.Stair(name,description,type,floor,building_name,bottomArea,numberRiser,numberTread,heightRiser,heightTread,isExternal,width,volume,material)

Q_IFC_Furniture(name,description,material,floor,room_name,building_name,bottomArea,volume,perimeter,height,width,depth,color):-
IFC.Furniture(name,description,material,floor,room_name,building_name,bottomArea,volume,perimeter,height,width,depth,color)

Q_IFC_Texture(name,repeatS,repeatT,tex_texture_type,world_to_texture,width,height,tex_image_url,texture_coordinates,tex_wrap_mode):-
IFC.Texture(name,repeatS,repeatT,tex_texture_type,world_to_texture,width,height,tex_image_url,texture_coordinates),tex_wrap_mode=wrap_mode(repeatS, repeatT)

Q_IFC_Material(name,layerThickness,isVentilated,type,weight,thermalIrEmissivity,visibleTransmittance,x3d_emissive_color,x3d_transparency):-
IFC.Material(name,layerThickness,isVentilated,type,weight,thermalIrEmissivity,visibleTransmittance),x3d_emissive_color=emissive(thermalIrEmissivity),x3d_transparency=transparency(visibleTransmittance)

GLAV for CityGML:
GAV:
Q_CityGML_Building(description,year_of_construction,roof_type,id,name,building_parent_id,building_root_id,class,function,usage,year_of_demolition,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,area,numStorey):-
M.Building(building_name,area,numStorey,fireRating,description,objectPlacement,elevationOfRefHeight,elevationOfTerrain,buildingAddress,projectIssueDate,floor_name_above,floor_number_above,floor_height_above,floor_type,floor_elevation_above,categoryType,floor_color,floor_name_below,floor_number_below,floor_height_below,floor_elevati

on_below,workSchedule_name,type,identifier,creationDate,creators,purpose,duration,totalFloat,year_of_construction,finishTime,workControlType,userDefinedControlType,roof_name_below,number_below,roof_type,roof_height_below,elevation_below,roof_color,categoryCode,id,name,building_parent_id,building_root_id,class,function,usage,year_of_demolition,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry)

Q_CityGML_BuildingAddress(building_name,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint,addressLines):-
M.BuildingAddress(building_name,purpose,description,addressLines,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint)

Q_CityGML_Wall(wall_name,description,wall_type,building_id,room_id,lodX_multi_surface_id,opening_id,building_name,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,room_name,is_solid,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,opening_name,name_codespace,type,address_id,area,perimeter,length,height):-
M.Wall(wall_name,description,building_name,floor,area,perimeter,height,length,thickness,volume,color,materialLayer,rooms,connectedWall,openingElements,isExternal,is_solid,wall_type,building_id,room_id,lodX_multi_surface_id,opening_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,room_name,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,opening_name,name_codespace,type,address_id,table_nmame,"geometry",diminfo,srid)

```
Q_CityGML_Door(name,description,opening_id,name_codespace,"door",address_id,lod
X_multi_surface_id,building_name,building_id,building_parent_id,building_root_id,clas
s,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,st
oreys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_height
s_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,is_solid
,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulate
d,geometry,thematic_surface_id,type,room_id,room_name,area,height,width):-
M.Door(id,name,description,door_type,building_name,room_name,operationType,floor,a
rea,height,width,thickness,material,is_solid,isExternal,fireRating,color,opening_id,name_
codespace,"door",address_id,lodX_multi_surface_id,building_id,building_parent_id,buil
ding_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,me
asured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_grou
nd,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geo
metry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_tr
iangulated,geometry,thematic_surface_id,type,room_id,table_nmame,"geometry",diminf
o,srid)

Q_CityGML_Room(description,room_name,name_codespace,class,function,usage,buildi
ng_id,lodX_geometry_id,building_name,building_parent_id,building_root_id,year_of_co
nstruction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys
_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrai
n_intersection,lodX_multi_curve,lodX_multi_surface_id,gmlid,gmlid_codespace,parent_
id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,area,perime
ter,length,width):-
M.Room(space_name,number,description,building_name,room_type,floor,area,height,wi
dth,length,perimeter,volume,hasWindow,interiorOrExteriorSpace,room_name,name_cod
espace,class,function,usage,building_id,lodX_geometry_id,building_parent_id,building_r
oot_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_abo
ve_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_g
round,lodX_terrain_intersection,lodX_multi_curve,lodX_multi_surface_id,gmlid,gmlid_
```

codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry)

Q_CityGML_Stair(name,description,isExternal,id,name_codespace,class,function,usage,building_id,room_id,lodX_geometry_id,building_name,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve):-
M.Stair(name,description,type,floor,building_name,bottomArea,numberRiser,numberTread,heightRiser,heightTread,isExternal,width,volume,material,id,name_codespace,class,function,usage,building_id,room_id,lodX_geometry_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve)

Q_CityGML_Furniture(name,description,name_codespace,class,function,usage,room_id,lodX_geometry_id,lodX_implicit_rep_id,lodX_implicit_ref_point,lodX_implicit_transformation,room_name,building_id,building_name,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,is_texture_parametrization,world_to_texture,texture_coordinates,surface_data_id,surface_data_name,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,bottomArea,perimeter,height):-
M.Furniture(name,description,material,floor,room_name,building_name,bottomArea,volume,perimeter,height,width,depth,color,name_codespace,class,function,usage,room_id,lodX_geometry_id,lodX_implicit_rep_id,lodX_implicit_ref_point,lodX_implicit_transformation,building_id,building_parent_id,building_root_id,year_of_construction,year_of_dem

olition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,is_texture_parametrization,world_to_texture,texture_coordinates,surface_data_id,surface_data_name,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,table_nmame,"geometry",diminfo,srid)

<u>Q_CityGML_Texture</u>(name,tex_texture_type,tex_image_url,surface_data_id,gmlid,gmlid_codespace,name_codespace,description,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image,tex_mime_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,world_to_texture,texture_coordinates,surface_geometry_id,is_texture_parametrization):-
M.Texture(name,repeatS,repeatT,tex_texture_type,world_to_texture,width,height,tex_image_url,texture_coordinates,surface_data_id,gmlid,gmlid_codespace,name_codespace,description,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image,tex_mime_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,surface_geometry_id,is_texture_parametrization)

<u>Q_CityGML_Material</u>(name,type,id,gmlid,gmlid_codespace,name_codespace,description,is_front,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point):-
M.Material(name,layerThickness,isVentilated,type,weight,thermalIrEmissivity,visibleTransmittance,id,gmlid,gmlid_codespace,name_codespace,description,is_front,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_em

issive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point)

LAV:

Q_CityGML_Building(description,year_of_construction,roof_type,id,name,building_parent_id,building_root_id,class,function,usage,year_of_demolition,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,area,numStorey):-
CityGML.Building(description,year_of_construction,roof_type,id,name,building_parent_id,building_root_id,class,function,usage,year_of_demolition,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id),CityGML.Surface_Geometry(lodX_geometry_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry),area=SDO_GEOM.SDO_AREA(geometry, 0.005),numStorey=storeys_above_ground+storeys_below_ground

Q_CityGML_BuildingAddress(building_name,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint,addressLines):-
CityGML.Address(building_name,postalBox,city,region,postalCode,country,street,house_number,xal_source,multipoint),addressLines=house_number+" "+street

Q_CityGML_Wall(wall_name,description,wall_type,building_id,room_id,lodX_multi_surface_id,opening_id,building_name,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,room_name,is_solid,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,opening_name,name_codespace,type,address_id,area,perimeter,length,height):-

CityGML.Thematic_Surface(wall_name,description,wall_type,building_id,room_id,lodX_multi_surface_id,opening_id),CityGML.Building(description,building_name,building_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id),CityGML.Room(description,building_id,room_id,class,function,usage,lodX_geometry_id,room_name),CityGML.Surface_Geometry(is_solid,lodX_multi_surface_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry),CityGML.Opening(description,lodX_multi_surface_id,opening_id,opening_name,name_codespace,type,address_id),area=SDO_GEOM.SDO_AREA(geometry, 0.005),perimeter=SDO_GEOM.SDO_LENGTH(geometry, diminfo),length=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 1),height=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 2)

Q_CityGML_Door(name,description,opening_id,name_codespace,"door",address_id,lodX_multi_surface_id,building_name,building_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id,is_solid,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry,thematic_surface_id,type,room_id,room_name,area,height,width):-
CityGML.Opening(name,description,opening_id,name_codespace,"door",address_id,lodX_multi_surface_id),CityGML.Building(description,building_name,building_id,building_parent_id,building_root_id,class,function,usage,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_geometry_id),CityGML.Surface_Geometry(is_solid,lodX_multi_surface_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_composite,is_triangulated,geometry),CityGML.Opening_to_them_surface(opening_id,thematic_surface_id),CityGML.Address_to_building(address_id,building_id),CityGML.Thematic_Surface(name,description,lodX_multi_surface_id,building_id,thematic_surface_id,type,room_id),CityGML

.Room(description,room_name,building_id,class,function,usage,lodX_geometry_id,room_id),area=SDO_GEOM.SDO_AREA(geometry, 0.005),height=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 1),width=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 2)

Q_CityGML_Room(description,room_name,name_codespace,class,function,usage,building_id,lodX_geometry_id,building_name,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve,lodX_multi_surface_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry,area,perimeter,length,width):-
CityGML.Room(description,room_name,name_codespace,class,function,usage,building_id,lodX_geometry_id),CityGML.Building(description,building_name,class,function,usage,building_id,lodX_geometry_id,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve),CityGML.Surface_Geometry(lodX_multi_surface_id,gmlid,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangulated,geometry),area=SDO_GEOM.SDO_AREA(geometry, 0.005),perimeter=SDO_GEOM.SDO_LENGTH(geometry),length=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry,DIMINFO,1),width=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry, diminfo, 2)

Q_CityGML_Stair(name,description,isExternal,id,name_codespace,class,function,usage,building_id,room_id,lodX_geometry_id,building_name,building_parent_id,building_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrain_intersection,lodX_multi_curve):-
CityGML.Building_Installation(name,description,isExternal,id,name_codespace,class,function,usage,building_id,room_id,lodX_geometry_id),CityGML.Building(description,bui

lding_name,class,function,usage,building_id,lodX_geometry_id,building_parent_id,build ing_root_id,year_of_construction,year_of_demolition,roof_type,measured_height,storeys _above_ground,storeys_below_ground,storey_heights_above_ground,storey_heights_bel ow_ground,lodX_terrain_intersection,lodX_multi_curve)

Q_CityGML_Furniture(name,description,name_codespace,class,function,usage,room_id, lodX_geometry_id,lodX_implicit_rep_id,lodX_implicit_ref_point,lodX_implicit_transfor mation,room_name,building_id,building_name,building_parent_id,building_root_id,year _of_construction,year_of_demolition,roof_type,measured_height,storeys_above_ground, storeys_below_ground,storey_heights_above_ground,storey_heights_below_ground,lod X_terrain_intersection,lodX_multi_curve,gmlid,gmlid_codespace,parent_id,root_id,is_xli nk,is_reverse,is_solid,is_composite,is_triangulated,geometry,is_texture_parametrization, world_to_texture,texture_coordinates,surface_data_id,surface_data_name,is_front,type,x 3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_c olor,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_te xture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_refere nce_point,bottomArea,perimeter,height):-
CityGML.Building_Furniture(name,description,name_codespace,class,function,usage,ro om_id,lodX_geometry_id,lodX_implicit_rep_id,lodX_implicit_ref_point,lodX_implicit_t ransformation),CityGML.Room(description,room_name,class,function,usage,room_id,lo dX_geometry_id,building_id),CityGML.Building(description,building_name,class,functi on,usage,lodX_geometry_id,building_id,building_parent_id,building_root_id,year_of_co nstruction,year_of_demolition,roof_type,measured_height,storeys_above_ground,storeys _below_ground,storey_heights_above_ground,storey_heights_below_ground,lodX_terrai n_intersection,lodX_multi_curve),CityGML.Surface_Geometry(lodX_geometry_id,gmli d,gmlid_codespace,parent_id,root_id,is_xlink,is_reverse,is_solid,is_composite,is_triangu lated,geometry),CityGML.TextureParam(lodX_geometry_id,is_texture_parametrization, world_to_texture,texture_coordinates,surface_data_id),CityGML.Surface_data(descriptio n,name_codespace,gmlid,gmlid_codespace,surface_data_id,surface_data_name,is_front,t ype,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diff use_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,t

ex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point),bottomArea=SDO_GEOM.SDO_AREA(geometry, 0.005),perimeter=SDO_GEOM.SDO_LENGTH(geometry),height=SDO_GEOM.SDO_MAX_MBR_ORDINATE(geometry,DIMINFO,1)

Q_CityGML_Texture(name,tex_texture_type,tex_image_url,surface_data_id,gmlid,gmlid_codespace,name_codespace,description,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image,tex_mime_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point,world_to_texture,texture_coordinates,surface_geometry_id,is_texture_parametrization):-
CityGML.Surface_data(name,tex_texture_type,tex_image_url,surface_data_id,gmlid,gmlid_codespace,name_codespace,description,is_front,type,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image,tex_mime_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point),CityGML.TextureParam(world_to_texture,texture_coordinates,surface_data_id,surface_geometry_id,is_texture_parametrization)

Q_CityGML_Material(name,type,id,gmlid,gmlid_codespace,name_codespace,description,is_front,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point):-
CityGML.Surface_data(name,type,id,gmlid,gmlid_codespace,name_codespace,description,is_front,x3d_shinness,x3d_transparency,x3d_ambient_intensity,x3d_specular_color,x3d_diffuse_color,x3d_emissive_color,x3d_is_smooth,tex_image_url,tex_image,tex_mime_type,tex_texture_type,tex_wrap_mode,tex_border_color,gt_prefer_worldfile,ft_orientation,ft_reference_point)