Enrichment of LiDAR Data With Semantic Data Attributes Based on Semantic WEB Technology

Linas Paskauskas

STUDENT:	Linas Paskauskas
STUDY NR:	20161156
PROGRAM:	MSc. Geoinformatics
SEMESTER:	4th semester
SUPERVISOR:	Carsten Keßler
LENGTH:	51
DELIVERY DATE:	June 8th 2018

Contents

List of Abbreviations

BIM	Building Information Model
FME	Feature Manipulation Engine
GeoJSON	Geography JavaScript Object Notation
GeoSPARQL	Geographic Query Language for RDF Data
GML	Geography Markup Language
HTML	HyperText Markup Language
IFC	Industry Foundation Classes
OGC	Open Geospatial Consortium
OSS	Open Source Software
OWL	Web Ontology Language
PDAL	Point Data Abstraction Library
RDF	Resource Description Framework
RDFS	RDF Schema
RIF	Rule Interchange Format
SPARQL	SPARQL Protocol and RDF Query Language
SRID	Spatial Reference System Identifier
SWRL	Semantic Web Rule Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

WEB	An Internet-based hypertext system
XML	Extensible Markup Language

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would firstly like to express my sincere gratitude to my supervisor Carsten Keßler for the continuous support for my master thesis. His motivation, advising and guidance helped me in research and writing of this thesis.

Besides my supervisor, I would like to thank my colleagues at COWI for support and sharing their professional experience. Special thanks to Morten Hertz Knudsen and Mevludin Mesha Besic for their professional consultations, support, and patience. Thanks to Rasmus Lindeneg Johansen and Jesper Falk for their support and motivation during the period of writing this thesis.

1 Introduction

Nowadays technologies evolve fast, and new techniques and technologies for data representation are continuously changing. In this research, new possibilities of spatial data representation in the Semantic Web will be explored.

The Semantic Web was established by Berners-Lee (Berners-Lee, Hendler, & Lassila, 2001). It is not new, and it functions as an extension of WEB (Berners-Lee, Hendler, & Lassila, 2001). With the growth of the amount of data on the Internet, the author understood how essentially important it had become to structure published data in a meaningful way. As current WEB consists of resources and links, the Semantic Web, as we can see from Figure 1 below, has types to describe those links and resources (Koivunen, M. R., & Miller, 2001). This way, the data was built in a more understandable

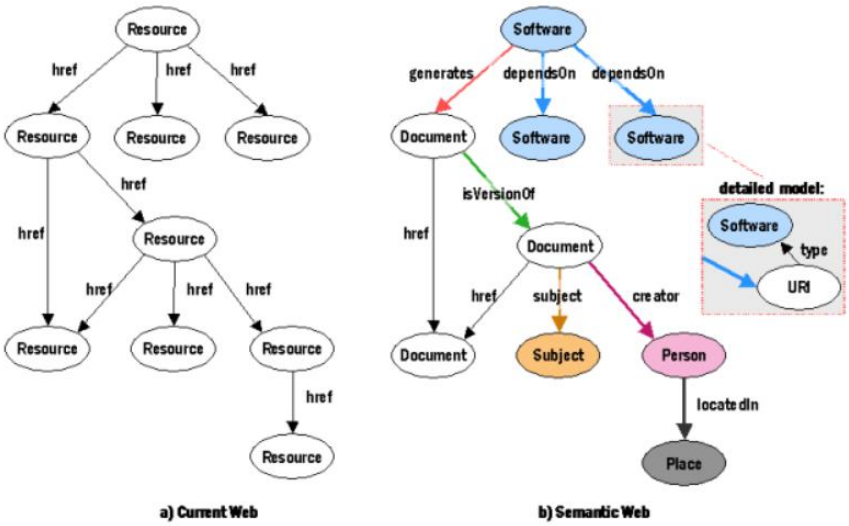

Figure 1. Current and Semantic Web examples (Koivunen, M. R., & Miller, 2001)

way or, in other words, in a machine-readable way. This can be achieved with the help of RDF and OWL languages. RDF syntax serves for source description and OWL adds relationships and semantics on top of it. Regarding this approach, each resource can be linked in between each other so that a computer or software can easily read and "understand" these relationships and published data.

In architectural industry, point clouds were widely used for different purposes such as generation of 3D models, reconstruction of indoor environments… (Ochmann, Vock, Wessel, & Klein, 2016). However, the approach mentioned above opens very interesting possibilities, which force us to explore them regarding publishing spatial data, in this case, indoor scanned point clouds into the Semantic Web. This could open new cases of use of indoor scanned data as for instance to discover relationship between two building elements or faster to export interesting semantic data attributes of a building. Several investigations have been done on how to extract geometry from point cloud or augment BIM model by using laser scanned data. These works will be explored in more detail later, in chapter 2.1 **Related Work**. However, regarding the correspondences with W3C, which are the main international Web standards and which were invented by Berners-Lee, there are no developed standards for point clouds intersection with semantics (World Wide Web Consortium, 2007).

The focus of this thesis is to investigate how laser scanned data can be enriched with semantic data attributes by using the Semantic Web technology. Additionally, the focus is to define the potential of used technology for building inspection purposes. Subsequently, the main research question of this thesis follows.

Research question: *How can LiDAR data be augmented with semantic data attributes?*

During the period of working on this thesis, a couple of sub-questions evolved. These questions are essential for further work. The success of augmented data depends on its capabilities across the Semantic Web.

- *What capabilities for published spatial data are there in the Semantic Web?*
- *How can the correct semantic annotations be identified for individual points/groups of points?*
- *What additional information can be extracted from the annotated dataset that is not already contained in the original point cloud?*

2 Theory

This chapter of the thesis is divided in two parts. The first part aims to look at and introduce related work done previously on point cloud labeling with semantics attributes, geometry recognition and work on spatial data integration into the Semantic Web. The second part aims to look at technologies of the Semantic Web. Furthermore, different formats as RDF and OWL will be described, as well as querying language SPARQL. Finally, the BIM format will be discussed.

2.1 Related Work

Some related research has been previously done on point cloud labelling with semantic data attributes as well as attempts to recognize and extract a geometry from a point cloud. In this section of the present chapter, some of those works will be presented.

As it was mentioned previously in the introduction, LiDAR is not a new format or a technique for collecting information about the surface of the Earth for analytical purposes. However, with the evolution of GIS technologies, LiDAR becomes more useful in construction industry.

Much research and testing was done in the past on indoors point clouds segmentation in order to reconstruct geometry or entire elements from point clouds. For instance, G. Antova, I. Kunchev, Ch.Mickrenska-Cheneva in their research “Point Cloud in BIM” present the benefits of point clouds for construction industry. They discussed point clouds applications for object-orientated modeling and

Figure 2. Point cloud (left) and 3D model (right)

integrating into BIM. Concerning paper point clouds, they can be used in different stages of a building lifecycle and can serve for 3D modelling of the designed object or enrich a BIM. 3D model can be created by using different methods such as surface triangulation or an object-wise. The method depends on software or external modelling application. Point clouds can be used to extract 4DBIM (BIM+time), 5DBIM (4DBIM+cost) and 6DBIM (5DBIM+facility management)(Antova, Kunchev, & Mickrenska-Cherneva, 2016). The mentioned paper did not answer my problem when it comes to point clouds labeling with semantics but helped to understand the interaction of BIM and point clouds.

The research "From Points Clouds to Building Information Models" done by H. Macher, T. Landes and P. Grussenmeyer presents the approach to creating a BIM model from point cloud. It is an relevant question pertaining old buildings designed a long time ago and which do not have a 3D model. Point clouds can help in a faster and more accurate 3D model comparison of models done from old 2D drawings. They developed their approach which consists of 2 parts. Bellow, the approach schema is presented in Figure 3.

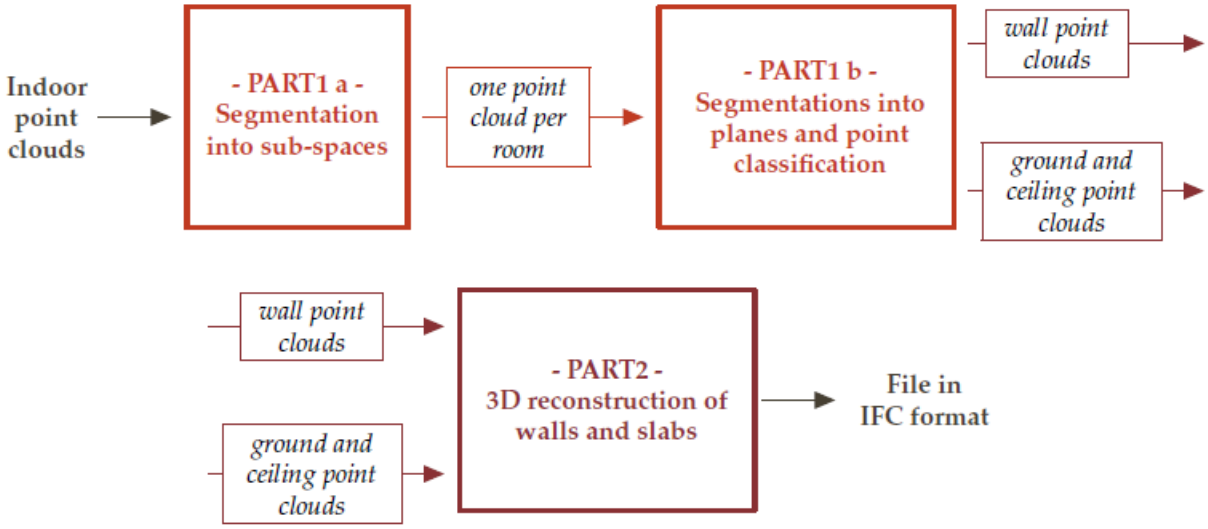

Figure 3. Overview of the developed approach (Macher, Landes, & Grussenmeyer, 2017)

The first part is the segmentation of point cloud into space and planes, and classification of points into several categories. The second part of the approach deals with reconstruction of a building elements such as walls, slabs etc., extracted from the point cloud in the first part of the approach.

In order to achieve better results, each part of the developed approach was split into several thresholds classified into the following categories:

- Thresholds related to spatial resampling of point clouds
- Thresholds related to constraints and quality criteria
- Thresholds related to space dimensions

The results of this project's first part of approach were promising from a geometrical and a semantic point of view. Sub-spaces were well recognized and segmented. Points are well classified into different categories. The second part of the approach has showed very satisfying results as well, since almost 90% of the input file objects were automatically reconstructed with the accuracy ranging from one millimeter to several centimeters.

Another project was "From 3D Point Clouds to Semantic Objects" made by H.B. Hmida, Ch. Cruz, F.Boochs and Ch. Nicolle. Their paper presents a knowledge-based approach using the OWL language, the Semantic Web Rule Language and 3D processing built-ins. Their focus was on 3D object detection and annotation in point clouds. More specifically, they use outdoor data (Deutsche Bahn) to detect different elements above the surface. WiDOP prototype, funded by the German government, was used for their approach. WiDOP provided with ontology structures installed behind it.

Another project, most closely related to the present one, is the master thesis "Semantic Enrichment of a Point Cloud based on an Octree for Multi-Storey Pathfinding" done by Florian W. Fichtner in 2016. In his thesis, he explored the usage of point clouds data and proposed a workflow for augmented point clouds usage for pathfinding.

He also tried to identify different building elements in point cloud and find walkable spaces for his human actor (Figure 4 below). With the help of Octree, a point cloud was labeled with the following building elements:

- stories
- floors as *walkable spaces*
- connections between stories, in this case restricted to stairs, as *walkable spaces*
- walls (static)
- obstacles (for example furniture, often not static)

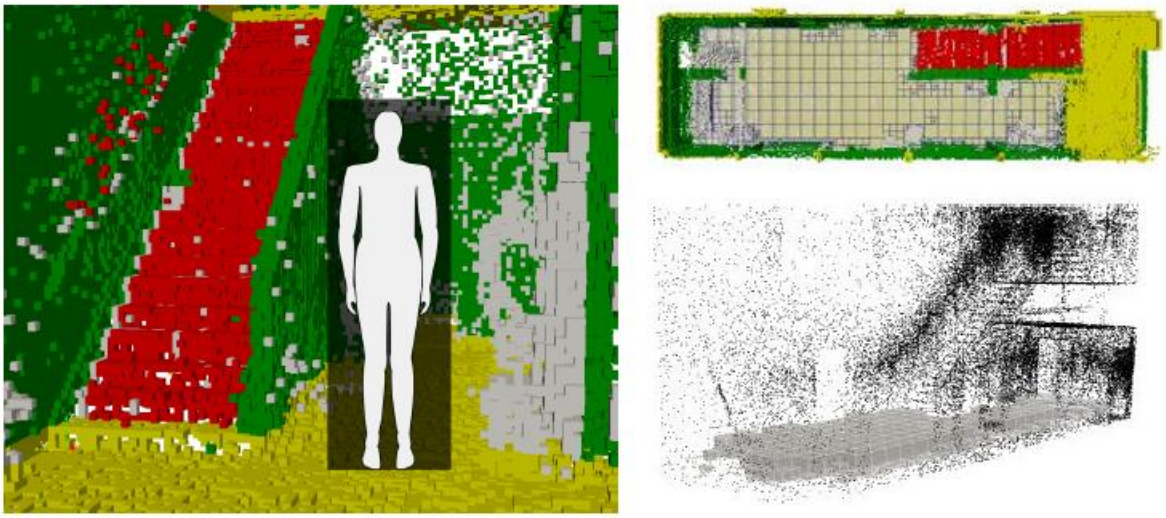

Figure 4. Semantically labeled point cloud for walkable path finding (Fichtner, 2016).

By identifying different building elements in point cloud and finding walkable spaces, many issues were highlighted when labeling different parts of a building. Most problematic elements turned out to be the ceiling, staircase and various obstacles. Ceiling divides a building into levels but it can cause confusion when detecting them. Staircase can be differentiated differently in the same software because it comes in different shapes. Obstacles are an issue for wall-detection because furnishings are close to a wall.

However, this research as well as the ones mentioned previously, were concerned more with the BIM model creation, augmentation or geometry detection by using laser scanned data. In these works, the idea of point clouds augmentation with semantic attributes, in order to use the Semantic Web method, was not considered.

So far, there has been no other work done on point clouds data integration into the Semantic Web. However, there exists done work on sensor special data integration into the Semantic Web.

2.2 RDF Overview

RDF – Resource Description Framework. RDF is an abstract data model which was designed by W3C working group. The first specification of RDF 1.0 was published by W3C in 2004, the RDF 1.1 in 2014 (Schreiber & Raimond, 2014). The Aim of RDF, as coded in its name, is to describe the structure of data. In other words, it is "data about data" or metadata. RDF is a standard model for data interchange on the Web (RDF working group, 2004). It produces data from machine-readable to machine-understandable (Lassila & Swick, 1999). Nowadays, when huge amounts of data are published on the Web, which keeps on growing every day, RDF helps to automate and interlink data to simplify Web searches for information and processing, both to Web users and developers.

RDF can be written in different languages, such as XML, turtle, n-triples – as it possesses interchangeable syntax. An example of a basic RDF syntax is showed below in Figure 5.

```
<rdf:RDF>
  <rdf:Description about="http://www.w3.org/Home/Lassila">
    <s:Creator>
      <rdf:Description about="http://www.w3.org/staffId/85740">
        <v:Name>Ora Lassila</v:Name>
        <v:Email>lassila@w3.org</v:Email>
      </rdf:Description>
    </s:Creator>
  </rdf:Description>
</rdf:RDF>
```

Figure 5. Example of RDF

Figure 5 can be translated from machine-understandable to a human-understandable with the help of a diagram. The XML converted into a diagram would look as follows (Figure 6 bellow).

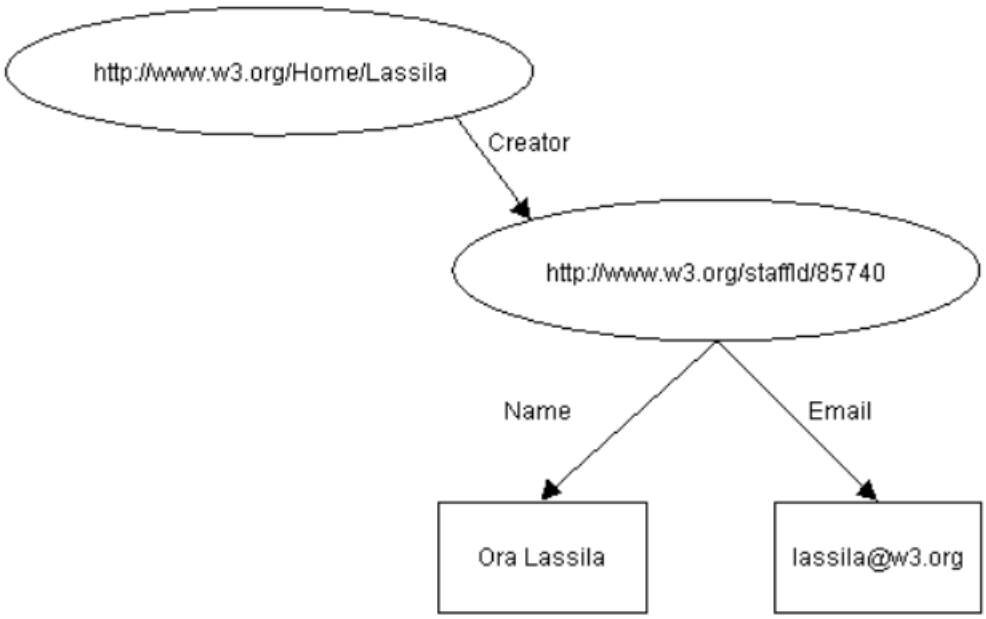

Figure 6. RDF diagram

RDF basic data model structure consists of three object types (Lassila & Swick, 1999):

- **Resource**

  All things that can be described by RDF expressions can be called Resources. For instance, a resource can be an entire Web page or just one element of it.

- **Properties**

  Properties are attributes or used relations to describe a resource.

- **Statements**

  Statement is a sum of a specific resource and a named property with the value of that property for that resource.

RDF format is widely used for different purposes. Most common cases of its use are for the description of time schedules, content description for search engines, electronic libraries, etc.

2.3 OWL Overview

OWL – Web Ontology Language is a Semantic Web language designed for the Semantic WEB and used on top of RDF to represent rich and complex knowledge about things, groups of things, and relations between things (OWL Working Group, 2012). It was developed by the Web Ontology Working Group as a part of the W3C Semantic Web activity and published on 10 February 2004 (Saha, 2007). Recent version is OWL 2.0.

Just as RDF, it was designed for Web application, but OWL was designed to process the content of the information, for instance by adding semantic attributes to it. However, unlike RDF, it was designed just to present the information to the user. OWL has a larger vocabulary which makes it easy to express different notions. It can also tell when two things in different schemas are the same thing.

Bellow in Figure 7 you can find small fragment of XML file (Link to the full XML file can be found in the Appendix) from Wine Ontology tutorial used in Protégé software.

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
  <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasMaker" />
  <owl:allValuesFrom rdf:resource="#Winery" />
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape" />
  <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
```

Figure 7. XML fragment of Wine ontology tutorial from Protégé

With the help of this software (as Protégé) all created ontologies can be visualized into hierarchy diagrams Figure 8 (Wine ontology, Protégé). These diagrams show all ontology elements and relationships between them in a more user-friendly and understandable way.

Figure 8. Wine Ontology Hierarchy Diagram

OWL, a more expressional language, is divided into three different sublanguages: OWL-Lite, OWL-DL, OWL-Full (Saha, 2007). As we can see from the Figure 8 below these sublanguages cover and supplement each other. Starting with OWL-Lite witch are subset of OWL-DL. OWL-Lite is

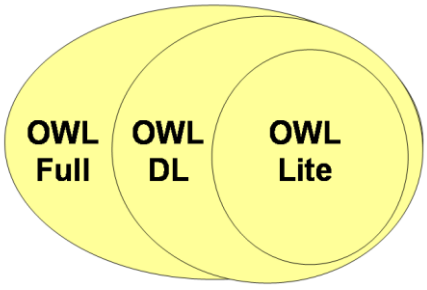

Figure 9. OWL Sublanguages ("OWL - the W3C Web Ontology Language," n.d.)

used for a simple ontology hierarchy structure. OWL-DL is a part of OWL-Full and has a maximum expressiveness while retaining a computational completeness. OWL-Full has a maximum

expressiveness and it provides a user with freedom of RDF modification with no computational guaranties.

Those sublanguages make up the OWL advance language and are useful for a wider range of communities, users, etc…

2.4 SPARQL & GeoSPARQL Overview

With the tremendous grow of semantic web within the World Wide Web Consortium proposed new query languages, SPARQL (Chebotko, Lu, Jamil, & Fotouhi, 2006). SPARQL is a query language for RDF, which allows for the specification of triple and graph patterns to be matched over RDF graphs (Chebotko et al., 2006). SPARQL and GeoSPARQL are based on OGC SimpleFeaturing, and they both are compatible with PostGIS. Since January 2008, SPARQL 1.0 has become an official W3C recommendation (*W3C Semantic Web Activity News – SPARQL*, 2008). Since March 2013, the current version has been SPARQL 1.1 (Harris & Seaborne, 2013). Along with new SPARQL version new features are available such as (July, 2009):

- **Subqueries** – Helpful in cases where it would be necessary to nest the results of a query within another query in a single SPARQL query.
- **Negation** – Negation by failure is possible as in SPARQL 1.0. Since it is difficult to write users asked SPARQL WG, hence the dedication language expressing negation.
- **Project Expressions** – Allow for SPARQL *SELECT* to project out any SPARQL expression rather than only variables bounded in a query.
- **Aggregations** – Aggregation functions allow users to query such functions as *COUNT*, *SUM*, *MIN* and *MAX*.
- **Property Path** – Allows for a more brief SPARQL expression for hierarchical structures graph patterns, with the ability to match arbitrary length paths.
- **Update** – Gives an ability to add, update or remove statements from a RDF graph.

- **Graph Store Protocol** – Allows for updates and manipulation of a RDF graph either in a plain Web server via HTTP or SPARQL end point.

A SPARQL query consists of the following parts, listed in the order as shown in (Feigenbaum, 2009):

- **Prefix Declaration** – *PREFIX foaf:* http://example.com/resource/
- **Dataset Definition** – *FROM* …
- **A Result Clause** – *SELECT* …
- **The Query Pattern** – *WHERE* …
- **Query Modifiers** – *ORDER BY* …

Figure 10 shows a simple SPARQL query. The examples are adopted from (Feigenbaum, 2009). The query is composed of three parts.

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    ?person foaf:name ?name .
}
```

Figure 10. A simple SPARQL query example

The first part is a prefix declaration *PREFIX foaf:* which shows abbreviating URIs. The second part is the *SELECT* clause which identifies the variables which appear in the query results. The third part is the *WHERE* clause which provides the basic graph, pattern to match against the data graph.

Expected Result

In order to achieve a more accurate result in more complex queries, *FILTERS* or *OPTIONAL* patterns can be used (Aliprand & Unicode Consortium, 2003).

Filters can help to restrict the set of solutions according to a given expression (Aliprand & Unicode Consortium., 2003). For instance, in the given example below *Figure 11*, adopted from (Wahid, Ahmad, Nor, & Rashid, 2017), landlocked countries can be filtered by specifying a population value greater than 15 million and filtering out the unwanted query results.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX type: <http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/>
SELECT ?country_name ?population
WHERE {
    ?country a type:LandlockedCountries ;
             rdfs:label ?country_name ;
             prop:populationEstimate ?population .
    FILTER (?population > 15000000) .
}
```

Figure 11. A FILTER query example

Expected Results

Optional patterns can be useful to have in queries where a piece of information for solution can be added if the information is available but if we do not want to reject the solution in case some parts of the information in the query patterns do not match (Aliprand & Unicode Consortium., 2003). This is because in the basic graph the entire query pattern must match since there has to be a solution. Bellow Figure 12 shows a query example adopted from (Wahid et al., 2017) where an *OPTIONAL* pattern in query is shown.

```
PREFIX mo: <http://purl.org/ontology/mo/>
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?name ?img ?hp ?loc
WHERE {
  ?a a mo:MusicArtist ;
     foaf:name ?name .
  OPTIONAL { ?a foaf:img ?img }
  OPTIONAL { ?a foaf:homepage ?hp }
  OPTIONAL { ?a foaf:based_near ?loc }
}
```

Figure 12. Example of OPTIONAL pattern in query

Expected Results

In solutions above Figure 12 *OPTIONAL* tries to match a graph pattern but the entire query will not fail if the optional match fails (Wahid et al., 2017).

For storing and querying geospatial information into Semantic Web GeoSPARQL comes in line.

GeoSPARQL was developed by OGC and it is a spatial extension of SPARQL. The main purpose of the GeoSPARQL standard is to store, query and to filter relationships between geo spatial entities on

the Semantic WEB by using RDF triples (Battle & Kolas, 2012). Considering the documentation (Matthew Perry & Herring, 2012), GeoSPARQL consists of three main features:

- An RDF/OWL vocabulary for representing spatial information consistent with the simple Features model
- A set of SPARQL extension functions for spatial computations
- A set of RIF rules for query transformation

Also GeoSPARQL has three main key classes (Kolas & Battle, 2012):

- `geo:Feature` – Is a thing that exists in a real world and can have a spatial location, for instance an airport, a hospital or a monument etc.;
- `geo:Geometry` – A representation of a spatial location, for instance a hospital can be presented as a point or as a detailed polygon;
- `geo:SpatialObject` – A superclass of both Features and Geometries.

BelowFigure 13 shows the relationship between these classes.

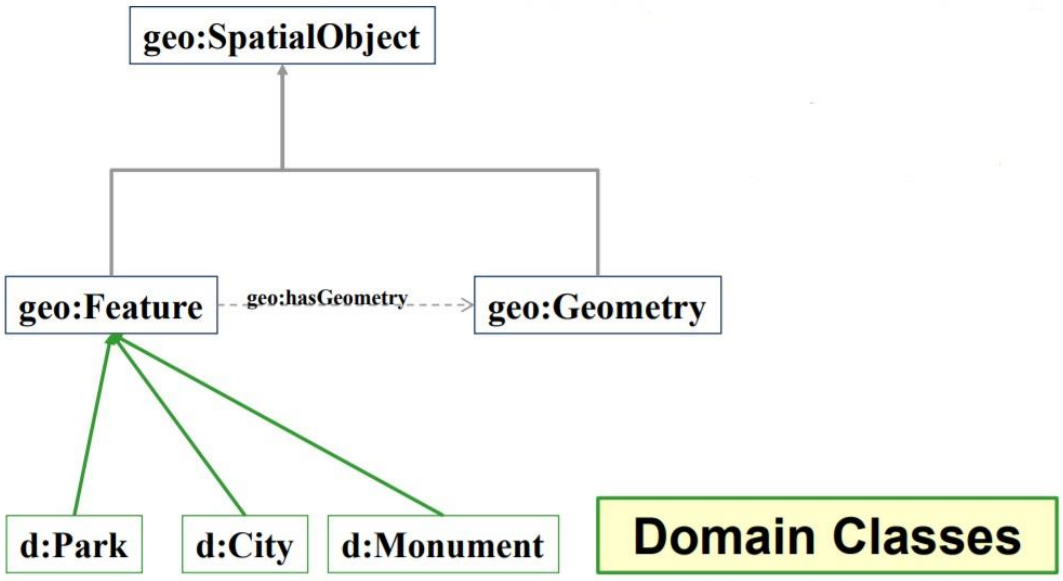

Figure 13. GeoSPARQL relationship between classes (Kolas, Perry, & Herring, 2012)

`Feature` and `Geometry` classes are the hooks for the users to plug in their own features and geometries in order to express linked geometric data.

There are two different ways to represent geometry literals and their associated type hierarchies (Battle & Kolas, 2011):

- **W**ell **K**nown **T**ext – SRID URI is added for `geo:wktLiteral`
- **G**eography **M**arkup **L**anguage – is used as-is for `geo:gmlLiteral`

Below, Table 1 shows WKT geometry types.

TYPE	SHAPE	Geometry Class	SYNTAX
POINT		`sf:Point`	`POINT(longitude latitude)`
LINESTRING		`sf:LineString`	`LINESTRING(long1 lat1, long2 lat2, …)`
POLYGON		`sf:Polygon`	`POLYGON((long1 lat1, long2 lat2, … , long1 lat1))`
POLYGON (WITH HOLE)		`sf:Polygon`	`POLYGON((long1 lat1, long2 lat2, … , long1 lat1), (longA latA, longB latB, …, longA latA))`

Table 1. WKT geometries (Kolas & Battle, 2012)

GeoSPARQL also provides the possibility to ask for topological relationships, such as overlaps between spatial entities (Battle & Kolas, 2012). This can be achieved in three possible ways (Kolas & Battle, 2012), by using:

1. GeoSPARQL filter functions
2. Geometry-to-geometry properties
3. Feature-to-feature properties

Improvements of SPARQL query language are being implemented. Users and developers both try to make it as good as possible for use in the Semantic Web.

GeoSPARQL as the extension of SPARQL is a powerful tool for representation, storing and querying of geospatial-linked data in the Semantic Web. With the help of this standard, such queries as the following become available (Matt Perry & Herring, 2013):

- Which Hotel within 20miles have appropriate treatment for my patient? (reasoning)

- What hotels with 3 star ratings are within 10km of at least 3 attractions with 4 star rating? (data integration)

2.5 BIM & IFC Overview

During the last few years BIM usage and its benefits have been widely discussed all around the world. However, it is still unclear for many people what BIM is and what exactly it does. Some people think that BIM is a software for modeling, others think that it is a basic 3D model, others still think that it is a virtual reality, etc…

BIM or Building Information Modeling is a process of creating and managing information of a building project across the project's lifecycle Figure 14 (National Building Specification, 2017). BIM comes in line in early building design-stage and continuously serves as an information source in each stage until the building is demolished.

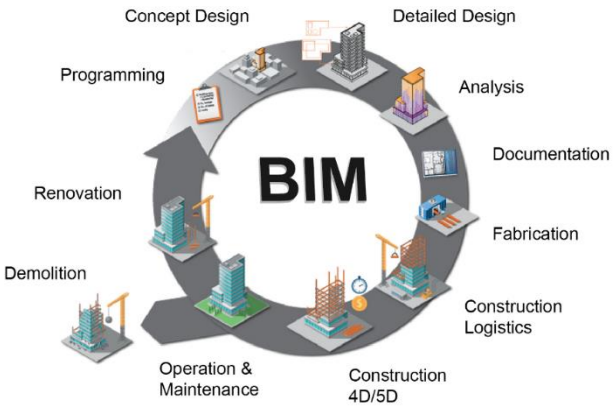

Figure 14. BIM usages diagram ("BIM," n.d.)

It allows a more efficient building construction process, seeing as BIM can provide engineers not only with 2D or 3D drawings but also with information about the amount of materials or the elements required during the recent stages of the process. That ensures a more efficient construction

process and saves both the materials and time. Nowadays, when we look increasingly for a sustainable design, all that makes BIM even more important and attractive.

When considering BIM, we cannot forget to mention the IFC or Industry Foundation Classes which were developed by Building SMART International. Ever since IFC4, it has been accepted as ISO-16739 standard (BuildSmart, 2014). It is a format which provides interoperability solutions between different types of software. IFC is capable of importing and exporting building objects and their properties and semantics. This way it helps to exchange information between architects, constructors, engineers, etc. A recent version of IFC4 Addendum 2 was published on 15th July 2016 as a buildingSMART Final Standart (BuildSmart). It covers all elements of a building, its physical (BuildSmart).

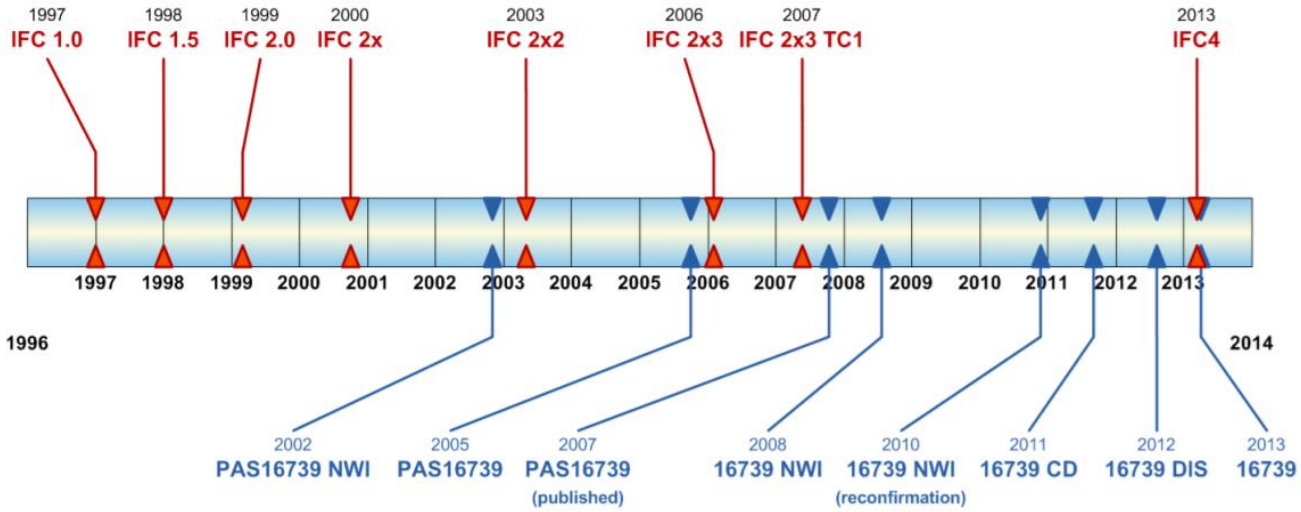

Figure 15. Versions of IFC format (BuildSmart, 2014).

IFC format does not cease to grow up and IFC5 is in its early stage of development.

3 Methodology

3.1 Introduction

This chapter of the thesis will describe the patch of developed workflow for a laser scanned indoor point cloud augmentation with semantic attributes. To establish the approach two datasets will be used. The datasets will be presented later in this chapter. This chapter also will link back to the Chapter **2.0** and clearly explain chosen techniques mentioned for approaching the developed workflow.

3.2 Data

COWI provided the data used in this thesis. The dataset consist of 3D point cloud data set *Figure 16* and Revit 3D model *Figure 17*.

Figure 16. 3D Laser scanned point cloud Figure

Figure 17. Revit 3D building model.

The point cloud scene Figure 16 was collected by using Trimble Sx10 Scanner and consisted of more than 699millions points which cover 9652.477m^2 outside area and 859,39 m² of indoor area. To scan the entire area approximately 150 scanning positions were used and it took five days to scan. Later separate files have been merged by using Trimble RealWork software. 3D Revit model was manually modeled from the point cloud by using Autodesk Revit software.

Due to the fact that processing LiDAR datasets is a heavy task for computers it was decided to use one building for the experiment which was cropped from the file. Cropped file Figure 18 bellow consists of more than 50 million points. It contains noise points which later will be deleted.

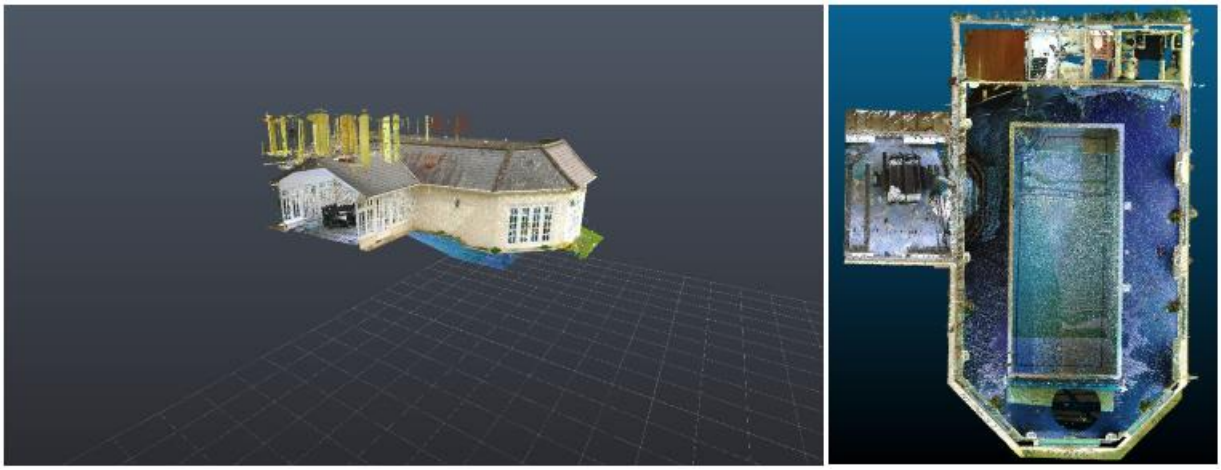

Figure 18. Building of for the investigation.

3.3 Workflow of Implementation

The main goal of this thesis is to investigate how laser scanned point cloud dataset can be augmented with semantic attributes by using the Semantic WEB technology. It means that from previously mentioned Revit 3D model semantic attributes will be extracted and they will be used for enrichment point cloud data set with these semantic attributes.

Since this method of point cloud usage is new it means that there is no relevant work done. There were similar works done on point cloud labeling with semantics in order to detect geometry, augment BIM models or for indoor navigation purposes. Relevant work observation was accomplished in order to become more familiar with methods of processing point cloud data, usage and possible approach of enriching them with semantic attributes.

After exploration of this field new framework for the workflow was constructed Figure 19. As the topic of this research is complex and work will include datasets in different formats and different tools will be used the Framework consists of three PARTS. Workflow begins from PART 1 "*Data Processing*" stage, and over in PART 3.

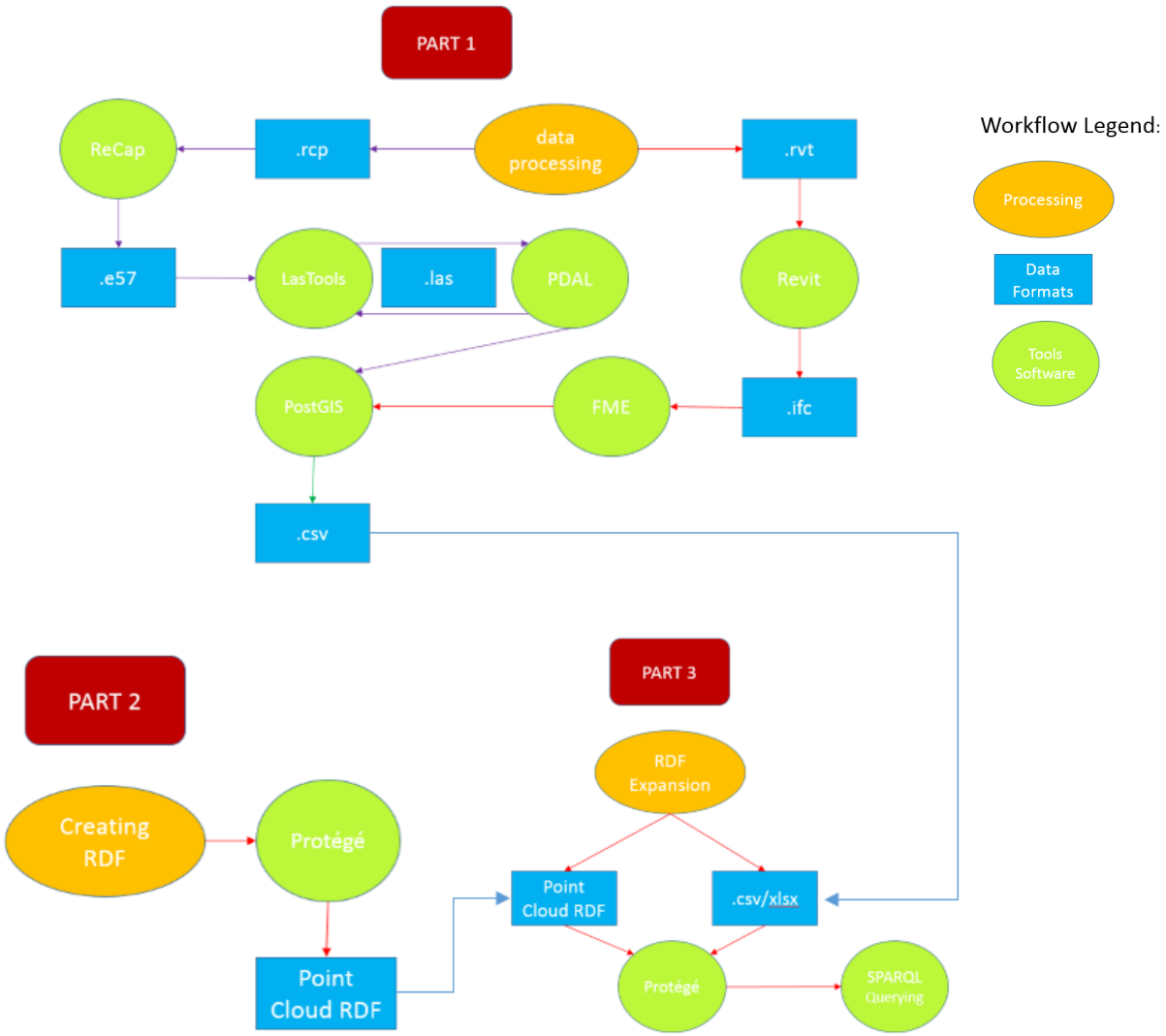

Figure 19. Workflow of the Implementation

Part 1 Figure 20 shows workflow done on datasets processing. As datasets arrived in Recap and Revit formats for the further processing they were translated into LAS and IFC formats. Different tools and software were used for processing and preparation for the experiment. After cleaning noise from point cloud and georeferencing of both files they were ready for the semantic enrichment. To extract the semantic attributes it was decided to use postGIS database. PostGIS database was chosen because it is powerful, easy and free to use tool. This part of work was done in order to extract semantic data attributes from the 3D Revit model and link them to the points from the point cloud.

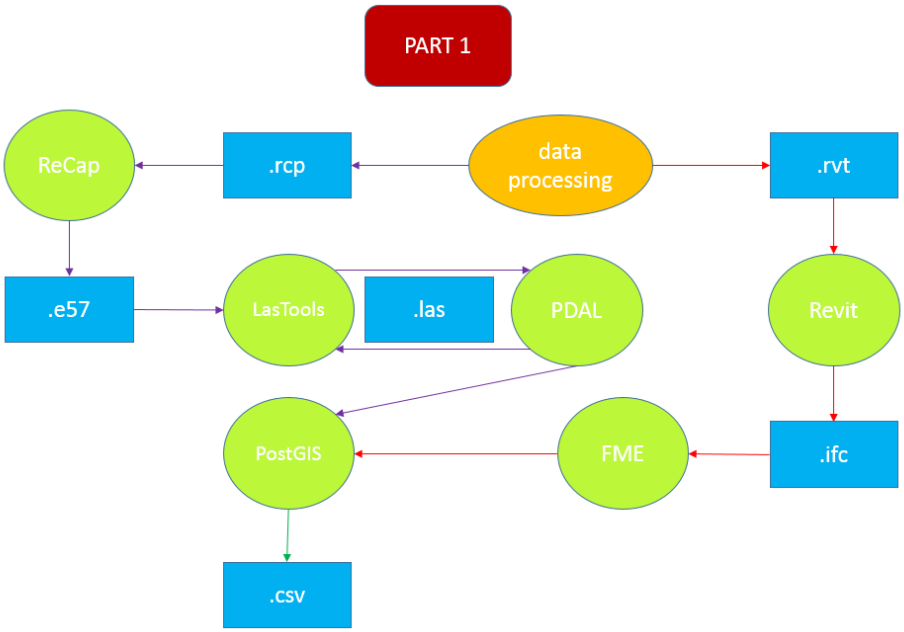

Figure 20.Part 1 of constructed Framework of the approach.

The goal of the PART 1 was to have ready data and the output file from a database with semantic attributes linked to the points. The Output file will be used further in PART3.

However, the use of semantic attributes for this approach requires all data to be structured in computer "understandable" way. This can be achieved by using RDF diagrams and OWL on top of RDF. These techniques were discussed in the 2.0 **Theory**.

PART 2 Figure 21 bellow, shows workflow done on developing RDF schema for point cloud data.

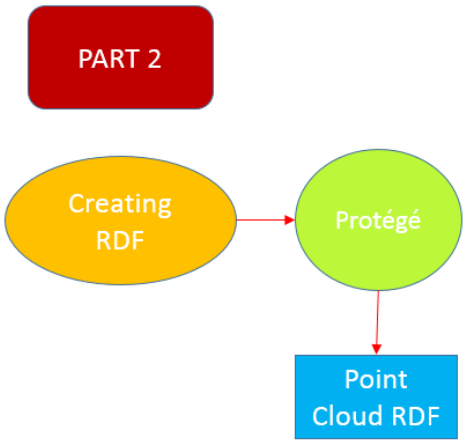

Figure 21.Part 2 of constructed Framework of the approach.

As can be seen from diagram *Figure 21* developing of RDF was accomplished with the help of Protégé software. By choosing one or another software it is importand to choose the one which fits user needs. Protégé was chosen rather than another software such as RaptorRDF, IsaViz, ontoStudio, etc because of it functionality and user friendly interface.

The last part of the workflow PART 3 *Figure 23* bellow, represents the last stage of the workflow where developed RDF diagram are expanded with OWL relationships and earlier in PART 1 of the

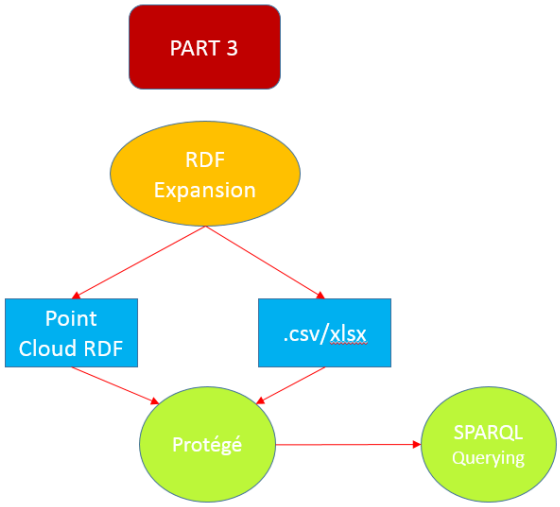

Figure 22. Part3 of constructed Framework of the approach.

workflow extracted semantic values are linked to the point cloud RDF diagram. Having prepared the point cloud RDF diagram is ready for testing in SPARQL. As SPARQL is buil-in in Protégé software all further work will be done in this software.

4 Implementation/Experimenting

4.1 Introduction

This chapter of the thesis aims to show in detail all the processes done to approach the workflow.

4.2 Data Preparation

Two data sets which have been presented in above **Chapter 3** are used for the experiment. It is worth to mention that as it is shown in **Chapter 3** not entire dataset will be used for the experiment but only the clipped one building from dataset Figure 23. As for data, preparation for the implementation nowadays is a natural and normal process and it is out of scope of this research it will not be presented in detail.

Figure 23. Left side 3D point cloud, right side Revit 3D model.

4.3 Revit Model Processing

For point cloud, annotation with semantics attributes will be used postGIS database. For this purpose, it is necessary to have both datasets exported into the previously mentioned database. For exporting 3D building into database Safe FME Desktop software was used. This software comes in separate packages such as:

- FME Data Inspector
- FME Workbench
- FME Quick Translator

It is a powerful tool which has overcome many of the problems related to data translation and can recognize almost all formats (FME, n.d.). Before starting to use FME, Revit model was converted to previously in **Chapter 2** descripted IFC format. This translation was done by using Revit. Translation was done because this format keeps 3D geometry, all semantic information and FME can read it and send it to the PostGIS database. For data export FME Workbench software was used. In this software new workspace was created Figure 24 for exporting IFC into the database. This workspace consists of four parts. The first part extracts geometry property from the input file, the second part extracts body geometry, the third part creates body surface, the last part projects the output into given projection in this case it is *EPSG: 27700 British National Grid* sends files into earlier created database and creates tables as well and geometry table.

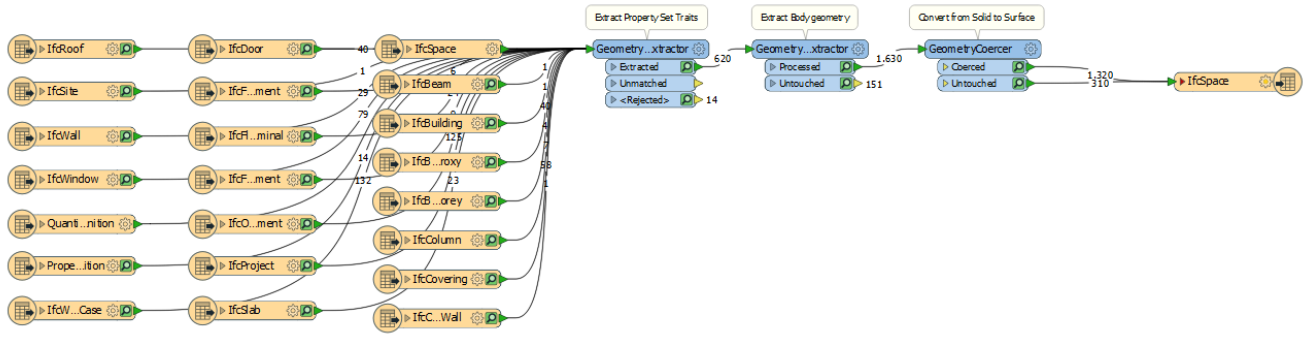

Figure 24. FME Workbench Workspace for exporting IFC into a PostGIS database.

After these steps were done the results were inspected by using FME Data Inspector tool. Results can be seen in Figure 25. As can be seen the model contains correct 3D geometry and in correct geographical projection *EPSG:27700.* As the results are positive since here work can be started on processing point cloud in the same database.

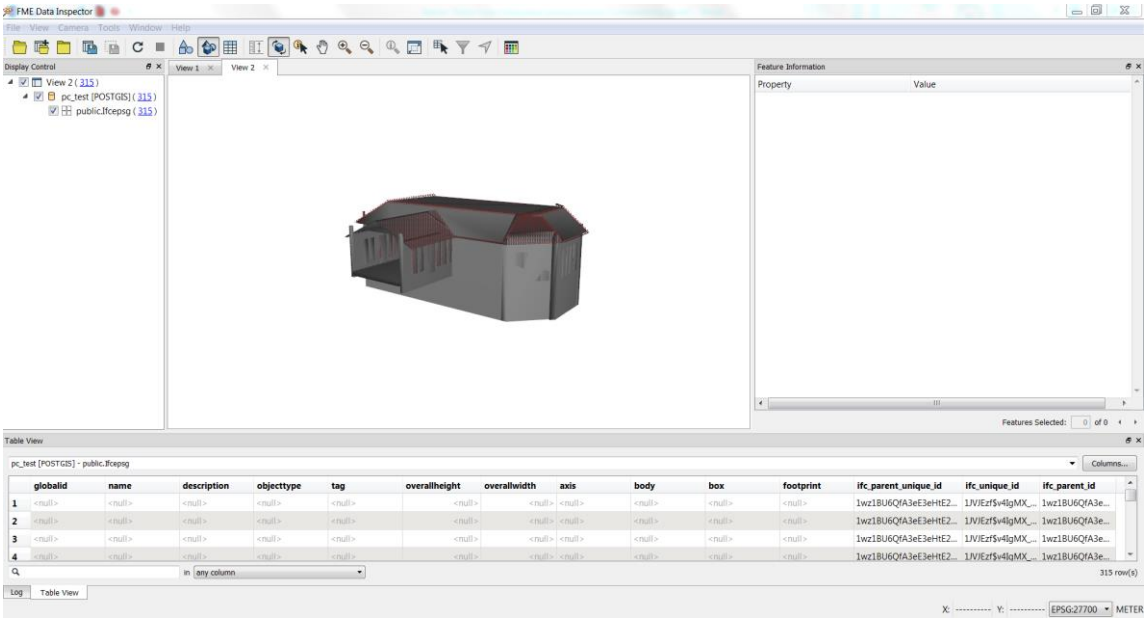

Figure 25. 3D Building model exposed from the PostGIS database.

4.4 Point Cloud Processing

Revit model point cloud has to be stored into the same database and has the same geographical reference system as Revit model. This is necessary for the further processing. For the point cloud, to store it into database PostgreSQL pointcloud extension will be used. Pointcloud extension allows to use point clouds with postGIS database (“Spatial relationships — OpenGeo Suite 4.6 User Manual,” n.d.). Two choices are available for storing point cloud in database. The first choice is to store each point as individual one by creating *PcPoint* and the second is to create patches *PcPatch* and store group of points in that patch. In order to avoid generating huge tablets which can be fiddly to manage, the second choice storing points into patches will be used for this research.

```
{
  "pipeline":[
    {
      "type":"readers.las",
      "filename":"C://MyTools/PDAL-master/test/data/las/building_EPSG.las",
      "spatialreference":"EPSG:27700"
    },
    {
      "type":"filters.chipper",
      "capacity":400
    },
    {
      "type":"writers.pgpointcloud",
      "connection":"host='localhost' dbname='pc_test' user='postgres' password='postgres' port='5432'",
      "table":"patches",
      "compression":"dimensional",
      "srid":"27700"
    }
  ]
}
```

Figure 26. PDAL pipeline for storing points as patches in the database.

For storing point cloud into database PDAL pipeline will be used. To be more specific, it is pointcloud pipeline Figure 26. The pipeline file is written in XML and can be divided into three main sections. The first section is reader ***readers.las*** that reads las file. Second section is writer ***writer.las*** that stores point cloud in specified database. In between there are filters ***filters.chipper***, which specify filter type and how many points will be stored in one patch.

Figure 27. Pipeline schema of the pointcloud extension ("Boundless : Analyzing and Visualizing LIDAR," n.d.).

After using above mentioned pipeline by using simple query was checked results Figure 28.

```
SELECT count(*) FROM patches;
```

Data Output	Explain	Messages	History

	count bigint
1	253024

```
WITH pts AS (
    SELECT PC_Explode(pa) AS pt
    FROM patches LIMIT 1
)
SELECT ST_AsEWKT(pt::geometry) FROM pts LIMIT 1;
```

	st_asewkt text
1	SRID=27700;POINT(-211600.41 -5527598.86 234.97)

Figure 28. Query for checking result of a point cloud storing into database.

In the left side it can be seen how many patches were created for storing points and the right side shows how point looks.

4.5 Point Cloud Labeling With Semantic Attributes

To accomplish the first part of developed workflow it remains to do points annotation with semantic attributes. As it was mentioned in the recent chapter previously point's annotation with semantics attributes will be succeeded in a database. For this purpose ST_3DIntersection function will be used. This function supports 3D and will not drop the z-index. It will return points that intersect with Revit model. Bellow Figure 29 shows code, which was used to retrieve results.

```
COPY (
WITH pts AS (
        SELECT PC_Explode(pa) AS pt
        FROM patches
)
SELECT ST_ASewkt(pt::geometry), *, ST_3DIntersects(
pts.pt::geometry, "Ifcepsg".geom)
FROM pts, "Ifcepsg"
LIMIT 500
)
to 'C:/MyTools/3D_Intersection.csv' WITH DELIMITER ',' csv HEADER;
```

Figure 29. 3DIntersection query.

This code can be divided into three parts. The first part explodes points from the patches. This is necessary in order to intersect each point as individual. The second part of code is a standard query for selecting functions and points of interest also describing from where to select and if necessary to set limits, in this case the limit is 500 points. The last part copies all results into csv file. This file as it was mentioned in the **Chapter 3** will be used for RDF expansion.

4.6 Designing Point Cloud RDF

Principles for designing RDF for the current point cloud will be the same as for all other RDF or systems to keep it simple and modular (Berners-Lee, 1999). The first principle will allow to easily understand it and the second ensures a possibility to integrate one RDF into another.

The current RDF for the point cloud will be created using Protégé software and will be divided into separate classes Figure 30.

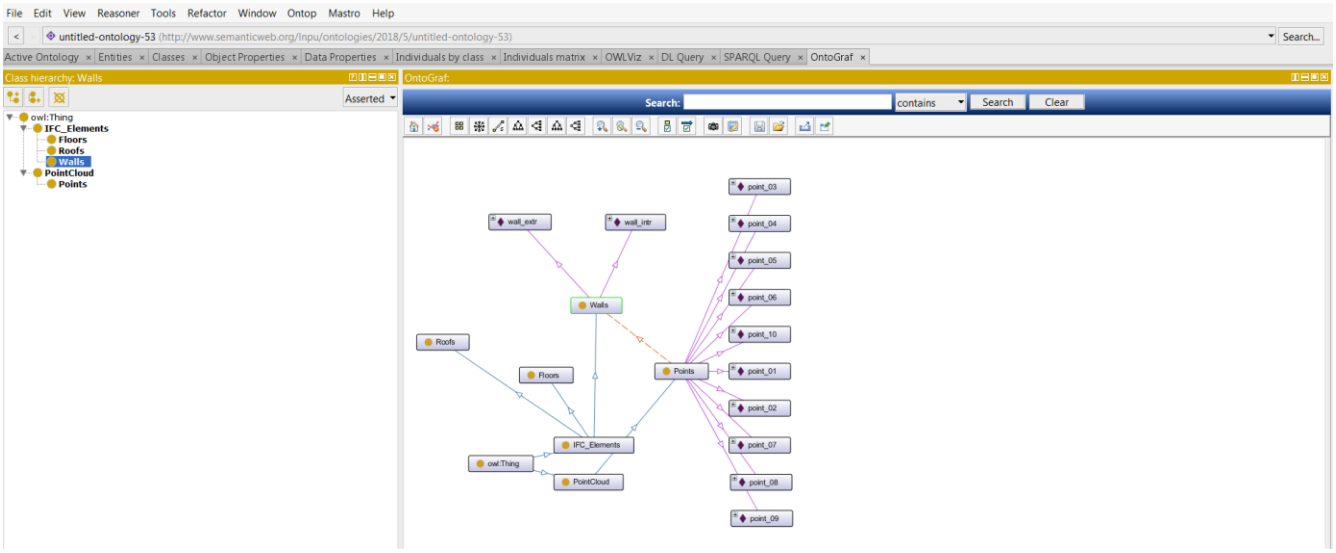

Figure 30. Point cloud RDF diagram.

Recent *Figure 30* represents created classes hierarchy on the left side and RDF diagram on the right side of the figure. Classes are sets, which contain individuals. In this case three classes ***IFC Elements***, ***Points*** are created and sub classes for each class. At the moment these classes are separated and do not have any relationship among each other. To ensure proper functionality relationships between classes will be created. This can be done by using three main types of properties such as (Prot et al., 2011):

- ***Object Properties*** – to explain relationship between two individuals or two instances
- ***Data Type Properties*** – to explain relationship between instances and data values
- ***Annotation Properties*** – to add metadata to class

For this experiment object properties and data type properties relationships will be used.

In order to extend RDF diagram with OWL complex relationships and link values to the classes Protégé build-in plugin Cellfie was used. This plugin is available from Protégé 5.x. It creates transformation rules from Excel spreadsheets that describe the OWL axiom structure and the tabular data. Rules are written in user-friendly Manchester Syntax Figure 31.

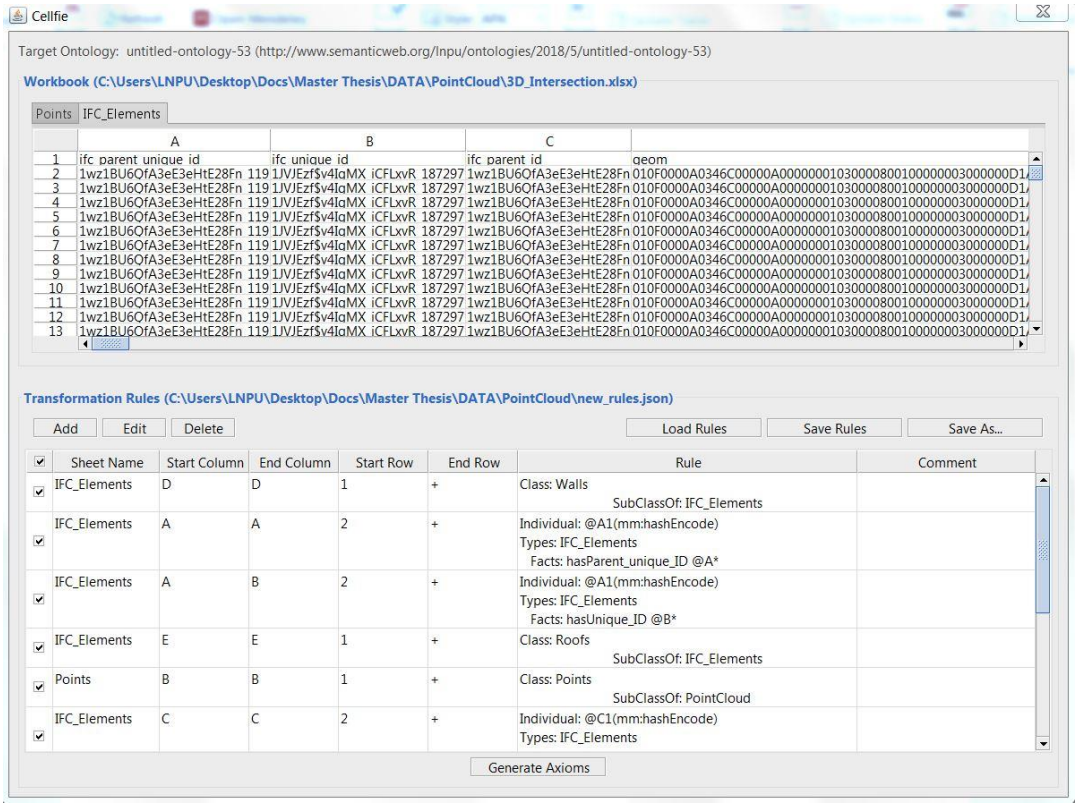

Figure 31. Cellfie window.

5 Discussion

In this section of the thesis research questions will be answered, possible further developing, and results will be discussed.

5.1 Research Questions

In this section research questions posted in **Chapter 1** will be answered. Firstly the sub questions and finally main question will be dealt with.

What capabilities for published spatial data are there in the Semantic Web?

After exploration of the Semantic Web technology it can be concluded that published spatial data capabilities are colossal. As recently published spatial data is heterogeneous, these datasets cannot communicate without data conversion (Zhang, Li, & Zhao, 2007). The Semantic Web technology can connect these datasets and make them quickly accessible.

How can the correct semantic annotations be identified for individual point/groups of points?

Sematic annotation for point clouds is a huge issue as the annotation of individual point would accumulate tremendous amount of data. This issue was noticed **Chapter 4** during this research. A more convenient way of dealing with point would be labeling as and storing group of points. This way accumulated data amount would be significantly reduced.

What additional information can be extracted from the annotated dataset that is not already contained in the original point cloud?

Semantic parsing of the information from annotated dataset depends on the structure of the semantic graph. This means that every additional information node can be reached and extracted from the dataset if it has a relatioship link with source node.

How can LiDAR data be augmented with semantic data attributes?

As it can be seen from the approach of this thesis and regarding **Chapter 2** for LiDAR data set augmentation several techniques can be used. The choice may depend on the needs of the user. For this thesis advance method of point's notation with semantic attributes was developed **Chapter 4**. Moreover, a new approach of usages augmented LiDAR data was shown.

5.2 Further Development

As this implementation of the point cloud method is new, a further and more diverse research should be devoted to it. This chapter of the thesis will take a look into what kind of research could be done in the future.

The recent approach requires that a large part of the processes should be done manually. In the future, efforts should be made in order to diminish the extent of manual work. That is because, depending on the operator skills and accuracy, manual work can cause different results while still using the same sets of data. Besides, automation processes could involve GeoSPARQL or such new formats as Potree in order to retrieve 3D visualizations of the results. During the work on this research, there arose an idea about using 3D building elements segmentation and hosting them in Potree format. Potree is OSS, which was funded by Georepublic, Veesus, Sigeom SA, Rapidlasso, the Ludwig Boltzmann Institute Archaological Prospection and Virtual Archeology and Pix4D (Schuetz, 2016). It is based on WebGL technology renderer for large point clouds (Chang & Lee, 2018). It is a powerful tool for point clouds rendering and analysis which, simultaneously, is easy to use.

The idea for using this tool is due to its ability to use binary geometry line to create 3D geometry of different parts of the Building Figure 32.

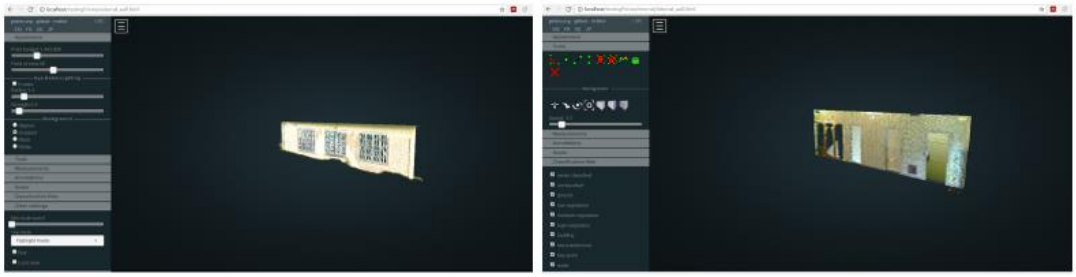

Figure 32. External wall in left and internal wall in right, converted in Potree format.

Potree has a OSS converter which builds octree from las, laz, binary ply, xyz or ptx files ("Potree Converter," n.d.). After retrieving SPARQL results in a geometry binary code which, by clicking it, could automatically activate Potree Converter to convert that code into Potree which, in turn, enables creating HTML file as output.

5.3 Results

This section of the thesis will present the results of the previously described implementation chapter (**Chapter 4)**. These results will be linked to the outlined main goals of the thesis.

As it was mentioned in the introduction chapter (**Chapter 1**) the main focus of this thesis is to investigate how laser scanned data can be enriched with semantic data attributes by using the Semantic Web technology. As well as to find out what the potential of used technology for building inspection purpose is. From here main three goals of the thesis can be outlined:

- *Point cloud enrichment with semantic atributes.*
- *Annotated point cloud integration into Semantic Web.*
- *Exploration of integrated point cloud potential for building inspection purpose.*

Point cloud enrichment with semantic atributes.

Enrichment was accomplished by using PostGIS database. As a result of this goal csv output file was generated.

Annotated point cloud integration into Semantic Web.

This goal was acomplished by using Protégé software. Firstly RDF graph was developed which was extended with previously from PostGIS database generated csv output file. This file extended RDF with more complex OWL relationship links and semantic values Figure 33. The full list of generated triples can be found in the appendix.

Complete List of Point Cloud Ontology Triples

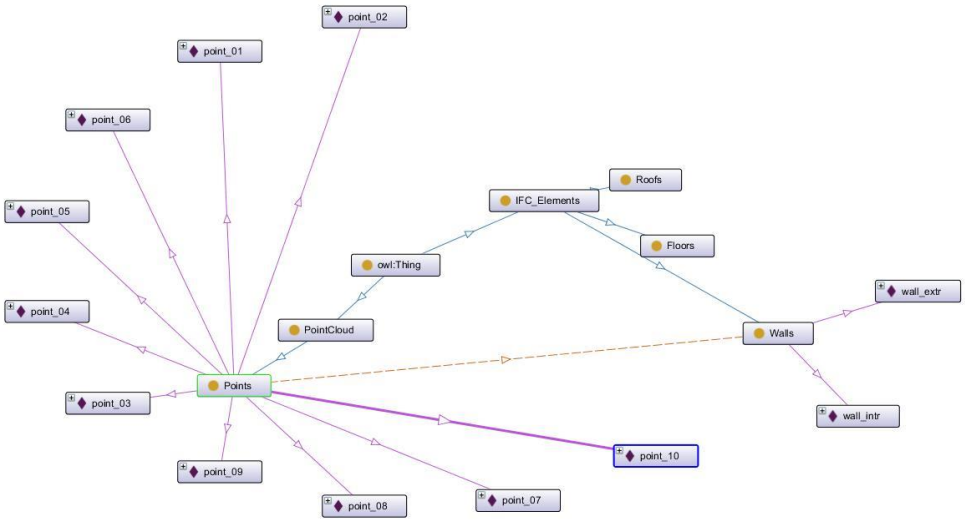

Figure 33. Point cloud ontology graph.

Exploration of integrated point cloud potential for building inspection purpose.

To explore the additional point cloud potential for building inspection purposes semantic parsing of developed point cloud semantic graph was done by using SPARQL querying plugin in Protégé software. Figure 34 shows SPARQL simple sample query which shows prefixes and simple *SELECT* query which selects all subjects and objects triples.

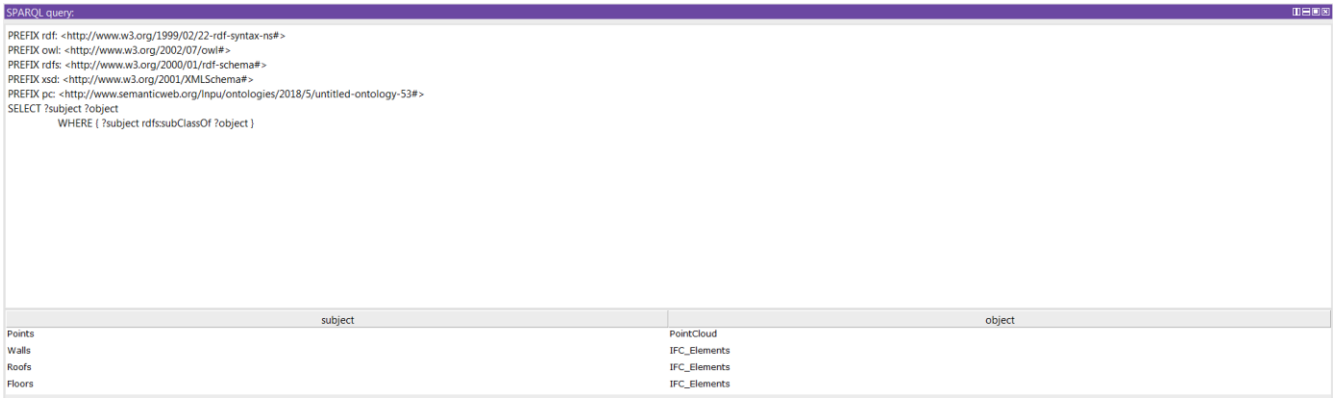

Figure 34. Simple SPARQL query.

Next Figure 35 shows simple SPARQL query. In the query as predicate is *Object Property* type *"intersect"* to ascertain which point what wall intersect.

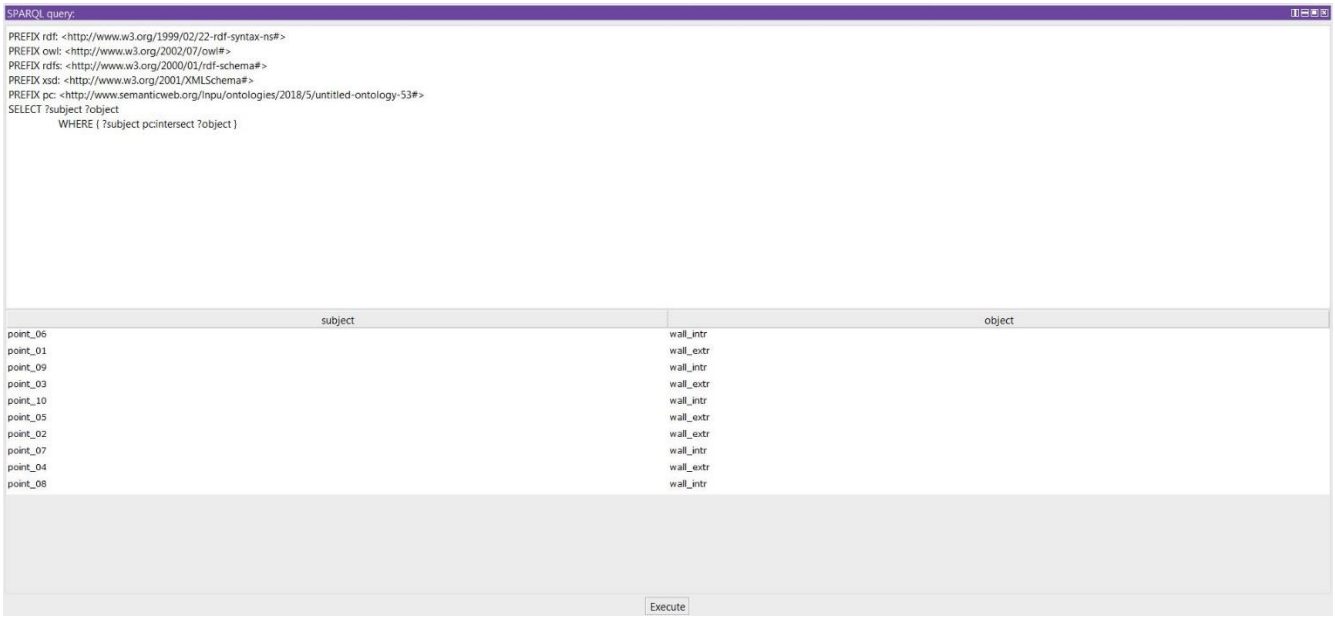

Figure 35. Simple SPARQL Object Property query example.

The last Figure 36 shows SPARQL query where as predicate is *Data Property* type “hasCoordinates” to ascertain what coordinates values has point instances.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX pc: <http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-53#>
SELECT ?subject ?object
        WHERE { ?subject pc:hasCoordinates ?object.}
```

subject	object
point_06	"SRID=27700;POINT(383988.437500057 398016.656249733 232.192687884556)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_08	"SRID=27700;POINT(383988.437500057 398016.656249733 232.212692156248)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_02	"SRID=27700;POINT(383988.96875 398004.468749995 234.943222044926)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_01	"SRID=27700;POINT(383988.999999998 398004.406250002 234.963363645624)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_07	"SRID=27700;POINT(383988.437500057 398016.656249733 232.202682391144)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_04	"SRID=27700;POINT(383988.96875 398004.468749995 234.943252566674)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_05	"SRID=27700;POINT(383988.999999998 398004.468749995 234.943267820733)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_10	"SRID=27700;POINT(383988.437500057 398016.687499733 232.202651871239)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_09	"SRID=27700;POINT(383988.437500057 398016.656249733 232.202697649661)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>
point_03	"SRID=27700;POINT(383988.96875 398004.468749995 234.95320129366)"^^<http://www.w3.org/2000/01/rdf-schema#Literal>

Figure 36. Simple SPARQL Data Property query example.

Since the tested ontology is primitive prototype, above shown simple SPARQL queries do not explore the entire potential of annotated point cloud for building inspection purposes. However, with ontology that is considerably more complex, SPARQL can answer more advance questions. Moreover, in the further must be considered GeoSPARQL usage for querying spatial geometries.

Further contributions would significantly increase point clouds potential for building purposes.

6 Conclusions

The above research shows a new contribution to point cloud data. As can be seen from **Chapter 2** this contribution is another look at usages of point cloud and is never tried or tested before. There is some similar work such as (H. Ben Hmida & Cruz, 2011) work testing WiDOP prototype with outdoor point cloud or (Fichtner, 2016) master thesis about indoor navigation. The reason why no work on point cloud integration in the Semantic Web has been done so far can be because of the limitation of processing point clouds. Large point cloud processing is a heavy task for computer. Moreover, storing these point clouds in database also has limitations linked to the capabilities of the system. Due to the last mentioned issue, the main point cloud was cropped and a smaller piece of it was used for the experiment. To avoid the recent issue in the future another method of point cloud annotation with semantics can be considered. However, the main goal of the thesis was successfully achieved. A workflow for point cloud augmentation with semantic attribute was developed **Chapter 3**. In **Chapter 4** the workflow was successfully tested and the point cloud was enriched with semantic information attributes. The results of enrichment were linked to previously designed primitive RDF schema and by doing that, OWL ontology for point cloud was developed. This means that point cloud was successfully augmented and integrated into the Semantic Web.

Due to the fact that point cloud data is a widely used format as based data for the further processing, any new contributions help to improve this data format and make it even more attractive.

7 Bibliography

Aliprand, J., & Unicode Consortium. (2003). *The Unicode standard*. Addison-Wesley. Retrieved from https://www.w3.org/TR/rdf-sparql-query/

Antova, G., Kunchev, I., & Mickrenska-Cherneva, C. (2016). Point clouds in BIM. *IOP Conference Series: Earth and Environmental Science*, *44*(4). https://doi.org/10.1088/1755-1315/44/4/042034

Battle, R., & Kolas, D. (2011). Linking geospatial data with GeoSPARQL. *Semantic Web*, *0*(0), 1–11.

Battle, R., & Kolas, D. (2012). GeoSPARQL: Enabling a Geospatial Semantic Web. *Semantic Web Journal*, *0*(0), 1–17. https://doi.org/10.3233/SW-2012-0065

Berners-Lee, T. (1999). Axioms of Web architecture. *World Wide Web Consortium*. Retrieved from https://www.w3.org/DesignIssues/Principles.html

BIM. (n.d.). Retrieved May 28, 2018, from http://www.charlesandrews.co.uk/bim.html

Boundless : Analyzing and Visualizing LIDAR. (n.d.). Retrieved May 21, 2018, from http://workshops.boundlessgeo.com/tutorial-lidar/

BuildSamrt. (n.d.). IFC4 Add2 Release — Welcome to buildingSMART-Tech.org. Retrieved March 29, 2018, from http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-add2

BuildSmart. (2014). IFC Overview summary — Welcome to buildingSMART-Tech.org. Retrieved March 13, 2018, from http://www.buildingsmart-tech.org/specifications/ifc-overview

Chang, T., & Lee, L. (2018). Automatic Monitoring System Based on IoT and Abstract :, (April), 1–16. https://doi.org/10.20944/preprints201804.0043.v1

Chebotko, A., Lu, S., Jamil, H. M., & Fotouhi, F. (2006). Semantics Preserving SPARQL-to-SQL Query Translation for Optional Graph Patterns. Retrieved from http://www.diag.uniroma1.it/~degiacom/didattica/semingsoft/SIS05-06/seminari-studenti/07-04-04 - SIS - Antonello Ercoli Alessandro Pezzullo - IBM Minerva/articoli/sparql/TR-DB-052006-CLJF.pdf

Feigenbaum, L. (2009). SPARQL By Example. *W3C*, 1–110. https://doi.org/10.1017/CBO9781107415324.004

Fichtner, F. W. (2016). Semantic enrichment of a point cloud based on an octree for multi-storey pathfinding,

(June). Retrieved from https://repository.tudelft.nl/islandora/object/uuid%3Ab4c13508-dbcc-4a32-a7b7-9cb22230e0a7

FME. (n.d.). What is FME? https://doi.org/10.1007/BF00044325

H. Ben Hmida, & Cruz, C. (2011). FROM 3D POINT CLOUDS TO SEMANTIC OBJECTS - An Ontology-based Detection Approach. *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*, (January), 255–260. https://doi.org/10.5220/0003660002550260

Harris, S., & Seaborne, A. (2013). SPARQL 1.1 Overview. Retrieved April 9, 2018, from https://www.w3.org/TR/sparql11-overview/

July, W. C. W. D. (2009). SPARQL New Features and Rationale This version : Latest version : Editors : *Language*, (July), 1–17. Retrieved from https://www.w3.org/TR/sparql-features/#Introduction_features

Koivunen, M. R., & Miller, E. (2001). W3C Semantic Web Activity. *W3C. Semantic Web Kick-Off in Finland*, 27–44. https://doi.org/10.1016/S0098-7913(03)00003-0

Kolas, D., & Battle, R. (2012). Geosparql user guide.

Kolas, D., Perry, M., & Herring, J. (2012). Getting Started with GeoSPARQL - Earth Science Ontology Dialog. Retrieved from http://ontolog.cim3.net/file/work/EarthScienceOntolog/2012-12-12_EarthScienceOntolog_session-5/GeoSPARQL_Getting_Started--DaveKolas_20121212.pdf

Lassila, O., & Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. https://doi.org/citeulike-article-id:122376

Macher, H., Landes, T., & Grussenmeyer, P. (2017). From Point Clouds to Building Information Models: 3D Semi-Automatic Reconstruction of Indoors of Existing Buildings. *Applied Sciences*, *7*(10), 1030. https://doi.org/10.3390/app7101030

National Building Specification, U. (2017). What is BIM? | NBS. Retrieved March 13, 2018, from https://www.thenbs.com/knowledge/what-is-building-information-modelling-bim

Ochmann, S., Vock, R., Wessel, R., & Klein, R. (2016). Automatic reconstruction of parametric building models from indoor point clouds. *Computers and Graphics (Pergamon)*, *54*, 94–103. https://doi.org/10.1016/j.cag.2015.07.008

OWL - the W3C Web Ontology Language. (n.d.). Retrieved June 1, 2018, from https://www.cs.vu.nl/~guus/talks/04-owl-brisbane/all.htm

OWL Working Group. (2012). OWL - Semantic Web Standards. Retrieved March 12, 2018, from https://www.w3.org/OWL/

Perry, M., & Herring, J. (2012). OGC GeoSPARQL-A geographic query language for RDF data. *OGC Candidate Implementation Standard*, 57. Retrieved from http://www.opengis.net/doc/IS/geosparql/1.0

Perry, M., & Herring, J. (2013). Getting Started with GeoSPARQL.

Potree Converter. (n.d.). Retrieved from https://github.com/potree/PotreeConverter

Prot, U., Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C., … Brandt, S. (2011). A Practical Guide To Building OWL Ontologies Using Protégè 4 and CO-ODE Tools Edition 1.3. *Matrix*, 0–107. Retrieved from http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

RDF working group. (2004). RDF - Semantic Web Standards. Retrieved March 11, 2018, from https://www.w3.org/RDF/

Saha, G. K. (2007). Web ontology language (OWL) and semantic web. *Ubiquity*, *2007*(September), 1–1. https://doi.org/10.1145/1295289.1295290

Schreiber, G., & Raimond, Y. (2014). Rdf 1.1 Primer. Retrieved March 12, 2018, from https://www.w3.org/TR/rdf11-primer/

Schuetz, M. (2016). Potree: Rendering Large Point Clouds in Web Browsers, 84.

Spatial relationships — OpenGeo Suite 4.6 User Manual. (n.d.). Retrieved May 21, 2018, from https://connect.boundlessgeo.com/docs/suite/4.6/dataadmin/pointcloud/postgis.html?highlight=pointcloud

Tim, B., Lee, B., Hendler, J., & Lassila, O. (2001). The Semantic Web will enable machines to COMPREHEND semantic documents, (May), 1–5.

W3C Semantic Web Activity News – SPARQL is a Recommendation. (2008). Retrieved from https://www.w3.org/blog/SW/2008/01/15/sparql_is_a_recommendation/

Wahid, H., Ahmad, S., Nor, M. A. M., & Rashid, M. A. (2017). Prestasi kecekapan pengurusan kewangan dan

agihan zakat: perbandingan antara majlis agama islam negeri di Malaysia. *Jurnal Ekonomi Malaysia*, *51*(2), 39–54. https://doi.org/10.1017/CBO9781107415324.004

World Wide Web Consortium. (2007). About W3C. Retrieved April 19, 2018, from https://www.w3.org/Consortium/

Zhang, C., Li, W., & Zhao, T. (2007). Geospatial data sharing based on geospatial semantic web technologies. *Journal of Spatial Science*, *52*(2), 35–49. https://doi.org/10.1080/14498596.2007.9635121

8 Appendix

3.2 OWL Overview

Link to Wine Ontology full XML file:

https://github.com/UCDavisLibrary/wine-ontology/blob/master/ontologies/www.example.org/wine.owl.xml

3.5 SPARQL & GeoSPARQL Overview

Link to the result of a simple SPARQL query

https://www.w3.org/2009/Talks/0615-qbe/#q1r (does not work with IE)

Link to the result of a FILTER pattern query example:

https://www.w3.org/2009/Talks/0615-qbe/#q5r (does not work with IE)

Link to the result of OPTIONAL pattern query example:

https://www.w3.org/2009/Talks/0615-qbe/#q7br (does not work with IE)

Point Cloud Ontology Triples:

```
@prefix : <http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-53#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-53> .

<http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-53>
rdf:type owl:Ontology .
```

```
#################################################################
#    Object Properties
#################################################################

###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#intersect
:intersect rdf:type owl:ObjectProperty ;
           rdfs:domain :Points ;
           rdfs:range :Walls .


#################################################################
#    Data properties
#################################################################

###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasCoordinates
:hasCoordinates rdf:type owl:DatatypeProperty ;
                rdfs:domain :Points ;
                rdfs:range rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasParent_ID
:hasParent_ID rdf:type owl:DatatypeProperty ;
              rdfs:domain :Walls ;
              rdfs:range rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasParent_unique_ID
:hasParent_unique_ID rdf:type owl:DatatypeProperty ;
                     rdfs:domain :Walls ;
                     rdfs:range rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasPoint_Geometry
:hasPoint_Geometry rdf:type owl:DatatypeProperty ;
                   rdfs:domain :Points ;
                   rdfs:range xsd:string .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasUnique_ID
:hasUnique_ID rdf:type owl:DatatypeProperty ;
              rdfs:domain :Walls ;
              rdfs:range rdfs:Literal .
```

```
###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasWall_Geometry
:hasWall_Geometry rdf:type owl:DatatypeProperty ;
                  rdfs:domain :Walls ;
                  rdfs:range rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#hasWall_Name
:hasWall_Name rdf:type owl:DatatypeProperty ;
              rdfs:domain :Walls ;
              rdfs:range rdfs:Literal .


#################################################################
#    Classes
#################################################################

###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#Floors
:Floors rdf:type owl:Class ;
        rdfs:subClassOf :IFC_Elements .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#IFC_Elements
:IFC_Elements rdf:type owl:Class .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#PointCloud
:PointCloud rdf:type owl:Class .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#Points
:Points rdf:type owl:Class ;
        rdfs:subClassOf :PointCloud .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#Roofs
:Roofs rdf:type owl:Class ;
       rdfs:subClassOf :IFC_Elements .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#Walls
:Walls rdf:type owl:Class ;
       rdfs:subClassOf :IFC_Elements .
```

```
#################################################################
#    Individuals
#################################################################

###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_01
:point_01 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_extr ;
          :hasCoordinates "SRID=27700;POINT(383988.999999998 398004.406250002
234.963363645624)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000DE0000000000000000000000000007B0078004300000000000000000C3F5B81E3BD4
09C10020AEA70B1655C1A4703D4AE1F26C40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_02
:point_02 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_extr ;
          :hasCoordinates "SRID=27700;POINT(383988.96875 398004.468749995
234.943222044926)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000D900000000000000000000000000006400630011000000000000000000C3F5B81E3BD4
09C10020AEA70B1655C17B14AE87EBF16C40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_03
:point_03 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_extr ;
          :hasCoordinates "SRID=27700;POINT(383988.96875 398004.468749995
234.95320129366)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000E2000000000000000000000000000038004C0007000000000000000000C3F5B81E3BD4
09C1F6480AA70B1655C10AD7A3B047F16C40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_04
:point_04 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_extr ;
          :hasCoordinates "SRID=27700;POINT(383988.96875 398004.468749995
234.943252566674)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000F1000000000000000000000000000061003F001200000000000000007B143E0A3BD4
09C1E19AC2A50B1655C15C8FC23533F36C40"^^rdfs:Literal .
```

```
###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_05
:point_05 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_extr ;
          :hasCoordinates "SRID=27700;POINT(383988.999999998 398004.468749995
234.943267820733)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000E8000000000000000000000000000006C005200170000000000000000AD733333BD4
09C1D7C31EA50B1655C15C8FC23533F36C40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_06
:point_06 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_intr ;
          :hasCoordinates "SRID=27700;POINT(383988.437500057 398016.656249733
232.192687884556)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000000000000000000000000000000006D006800480000000000000009A9951B886D4
09C10020AEB7081655C1EC51B83E0A076D40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_07
:point_07 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_intr ;
          :hasCoordinates "SRID=27700;POINT(383988.437500057 398016.656249733
232.202682391144)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000270000000000000000000000000013001D001200000000000000003333EB5186D4
09C1F6480AB7081655C133333353B8066D40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_08
:point_08 rdf:type owl:NamedIndividual ,
                  :Points ;
          :intersect :wall_intr ;
          :hasCoordinates "SRID=27700;POINT(383988.437500057 398016.656249733
232.212692156248)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000190000000000000000000000000000160020001800000000000000EC51703D86D4
09C1F6480AB7081655C133333353B8066D40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_09
:point_09 rdf:type owl:NamedIndividual ,
                  :Points ;
```

```
          :intersect :wall_intr ;
          :hasCoordinates "SRID=27700;POINT(383988.437500057 398016.656249733
232.202697649661)"^^rdfs:Literal ;
          :hasPoint_Geometry
"01010000001D0000000000000000000000000190023001A0000000000000000EC51703D86D4
09C1F6480AB7081655C133333353B8066D40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#point_10
:point_10 rdf:type owl:NamedIndividual ,
                    :Points ;
          :intersect :wall_intr ;
          :hasCoordinates "SRID=27700;POINT(383988.437500057 398016.687499733
232.202651871239)"^^rdfs:Literal ;
          :hasPoint_Geometry
"0101000000000000000000000000000000000006A0068003F0000000000000009A9951B886D4
09C1F6480AB7081655C15C8FC215AE076D40"^^rdfs:Literal .


###  http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-
53#wall_extr
:wall_extr rdf:type owl:NamedIndividual ,
                     :Walls ;
           :hasParent_ID "1wz1BU6QfA3eE3eHtE28Fn"^^rdfs:Literal ;
           :hasParent_unique_ID "1wz1BU6QfA3eE3eHtE28Fn_119"^^rdfs:Literal ;
           :hasUnique_ID "1JVJEzf$v4IgMX_iCFLxvR_187297"^^rdfs:Literal ;
           :hasWall_Geometry
"010F0000A0346C00000A0000000103000080010000003000000D1A6374F0863D040A236750A6F
967140FEFFFFFFFFAF83C0D1A6374F0863D040AA36750A6F967140FEFFFFFFFFAF83C0D1A6374F0
863D040A236750A6F967140FEFFFFFFFFAF83C00103000080010000000600000D1A6374F0863D0
40AA36750A6F9671408C259D1783C3A3C00633C2057B61D040E0CF92FADC1379408C259D1783C3A
3C00633C2057B61D040E0CF92FADC137940FEFFFFFFFFAF83C00633C2057B61D040CECF92FADC13
7940FEFFFFFFFFAF83C0D1A6374F0863D040AA36750A6F967140FEFFFFFFFFAF83C0D1A6374F086
3D040AA36750A6F9671408C259D1783C3A3C00103000080010000000600000001030EBDAB4DFD040
54964E570541C2C08C259D1783C3A3C0D1A6374F0863D040AA36750A6F9671408C259D1783C3A3C
0D1A6374F0863D040AA36750A6F967140FEFFFFFFFFAF83C0D1A6374F0863D040A236750A6F9671
40FEFFFFFFFFAF83C01030EBDAB4DFD04054964E570541C2C0FEFFFFFFFFAF83C01030EBDAB4DFD
04054964E570541C2C08C259D1783C3A3C00103000080010000000500000DCAB642442E1D0405A
296BC7F07CC2C08C259D1783C3A3C01030EBDAB4DFD04054964E570541C2C08C259D1783C3A3C01
030EBDAB4DFD04054964E570541C2C0FEFFFFFFFFAF83C0DCAB642442E1D0405A296BC7F07CC2C0
FEFFFFFFFFAF83C0DCAB642442E1D0405A296BC7F07CC2C08C259D1783C3A3C0010300008001000
000040000000633C2057B61D040E0CF92FADC137940FEFFFFFFFFAF83C03DCA43298543D0405001
0E913CB17840FEFFFFFFFFAF83C00633C2057B61D040CECF92FADC137940FEFFFFFFFFAF83C0063
3C2057B61D040E0CF92FADC137940FEFFFFFFFFAF83C00103000080010000000A0000001030EBDA
B4DFD04054964E570541C2C0FEFFFFFFFFAF83C0D1A6374F0863D040A236750A6F967140FEFFFFF
FFFAF83C0D1A6374F0863D040AA36750A6F967140FEFFFFFFFFAF83C00633C2057B61D040CECF92
FADC137940FEFFFFFFFFAF83C03DCA43298543D04050010E913CB17840FEFFFFFFFFAF83C0832B3
9294CC3D040E1BD4B2A0180C2C0FEFFFFFFFFAF83C0832B39294CC3D040E1BD4B2A0180C2C0FEFF
FFFFFFAF83C0862B39294CC3D040E1BD4B2A0180C2C0FEFFFFFFFFAF83C0DCAB642442E1D0405A2
96BC7F07CC2C0FEFFFFFFFFAF83C01030EBDAB4DFD04054964E570541C2C0FEFFFFFFFFAF83C001
0300008001000000070000000D1A6374F0863D040AA36750A6F9671408C259D1783C3A3C01030EBD
```

AB4DFD04054964E570541C2C08C259D1783C3A3C0DCAB642442E1D0405A296BC7F07CC2C08C259D
1783C3A3C0832B39294CC3D040E1BD4B2A0180C2C08C259D1783C3A3C03DCA43298543D04050010
E913CB178408C259D1783C3A3C00633C2057B61D040E0CF92FADC1379408C259D1783C3A3C0D1A6
374F0863D040AA36750A6F9671408C259D1783C3A3C00103000080010000000700000832B39294
CC3D040E1BD4B2A0180C2C0FEFFFFFFFFAF83C0832B39294CC3D040E1BD4B2A0180C2C08C259D17
83C3A3C0DCAB642442E1D0405A296BC7F07CC2C08C259D1783C3A3C0DCAB642442E1D0405A296BC
7F07CC2C0FEFFFFFFFFAF83C0862B39294CC3D040E1BD4B2A0180C2C0FEFFFFFFFFAF83C0832B39
294CC3D040E1BD4B2A0180C2C0FEFFFFFFFFAF83C0832B39294CC3D040E1BD4B2A0180C2C0FEFFF
FFFFFAF83C00103000080010000000500000832B39294CC3D040E1BD4B2A0180C2C0FEFFFFFFFF
AF83C03DCA43298543D04050010E913CB17840FEFFFFFFFFAF83C03DCA43298543D04050010E913
CB178408C259D1783C3A3C0832B39294CC3D040E1BD4B2A0180C2C08C259D1783C3A3C0832B3929
4CC3D040E1BD4B2A0180C2C0FEFFFFFFFFAF83C00103000080010000000500000000633C2057B61D
040E0CF92FADC137940FEFFFFFFFFAF83C00633C2057B61D040E0CF92FADC1379408C259D1783C3
A3C03DCA43298543D04050010E913CB178408C259D1783C3A3C03DCA43298543D04050010E913CB
17840FEFFFFFFFFAF83C00633C2057B61D040E0CF92FADC137940FEFFFFFFFFAF83C0"^^rdfs:Li
teral .

http://www.semanticweb.org/lnpu/ontologies/2018/5/untitled-ontology-53#wall_intr
:wall_intr rdf:type owl:NamedIndividual ,
                    :Walls ;
           :hasParent_ID "2UzHp$dq17VudTkAADZCu2"^^rdfs:Literal ;
           :hasParent_unique_ID "2UzHp$dq17VudTkAADZCu2_187535"^^rdfs:Literal ;
           :hasUnique_ID "2UzHp$dq17VudTkA2DZCu2_187566"^^rdfs:Literal ;
           :hasWall_Geometry
"010F0000A0346C000005000000010300008001000000040000000F27F63839596D440E0814D8211
AAA1C055B6602F06C794C0C44B004EF572D440D402FA42EEF57E40AAB7602F065395C0ECC1C130B
F86D0407285A188FD58A3C086DE132F06C794C0F27F63839596D440E0814D8211AAA1C055B6602F
06C794C00103000080010000000500000000F27F63839596D440E0814D8211AAA1C055B6602F06C79
4C0ECC1C130BF86D0407285A188FD58A3C086DE132F06C794C0CE8FC130BF86D040379AA188FD58
A3C086DE132F066F97C0D44D63839596D440A5964D8211AAA1C056B6602F066F97C0F27F6383959
6D440E0814D8211AAA1C055B6602F06C794C00103000080010000000500000000ECC1C130BF86D040
7285A188FD58A3C086DE132F06C794C0C44B004EF572D440D402FA42EEF57E40AAB7602F065395C
0A619004EF572D440AD5CF942EEF57E40AAB7602F06FB97C0CE8FC130BF86D040379AA188FD58A3
C086DE132F066F97C0ECC1C130BF86D0407285A188FD58A3C086DE132F06C794C00103000080010
0000005000000C44B004EF572D440D402FA42EEF57E40AAB7602F065395C0F27F63839596D440E0
814D8211AAA1C055B6602F06C794C0D44D63839596D440A5964D8211AAA1C056B6602F066F97C0A
619004EF572D440AD5CF942EEF57E40AAB7602F06FB97C0C44B004EF572D440D402FA42EEF57E40
AAB7602F065395C00103000080010000000400000D44D63839596D440A5964D8211AAA1C056B66
02F066F97C0CE8FC130BF86D040379AA188FD58A3C086DE132F066F97C0A619004EF572D440AD5C
F942EEF57E40AAB7602F06FB97C0D44D63839596D440A5964D8211AAA1C056B6602F066F97C0"^^
rdfs:Literal .

###
General axioms
###

[rdf:type owl:AllDisjointClasses ;
  owl:members (:Floors
                :Points

```
                :Roofs
                :Walls
              )
] .


###  Generated by the OWL API (version 4.2.8.20170104-2310)
https://github.com/owlcs/owlapi
```