Supporting the Integration of Building Data and Geospatial Data

Joseph O'Donovan

Supervised by Prof. Declan O'Sullivan & Dr. Kris McGlinn

A Dissertation submitted to the University of Dublin, Trinity College in fulfillment of the requirements for the degree of Integrated Masters in Computer Engineering

Submitted to the University of Dublin, Trinity College, April 2019

Declaration

I, Joseph O'Donovan, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature _____

Date _____

Summary

This research project explores methods of integrating Building Information Modelling (BIM) buildings with their corresponding Geographic Information Systems (GIS) buildings. Different geometric representations and coordinate systems make moving between the two domains a challenge. The difficulties in representing a BIM building as GIS motivate the research carried out by this project.

The current state of the art of BIM and GIS is first examined to discover why a standard process of BIM and GIS integration has not yet been found. The reasons for this set up the design of the solutions provided by this research project, where the natural openness of Linked Data is utilized in an attempt to provide a solution.

Four BIM and GIS integration solutions are provided by this project as part of a "solutions toolkit". This is a set of implemented solutions that attempt to solve the BIM and GIS integration problems by focusing on the geometric representation of a building. Each of these solutions interact with the ifcOWL format, a Linked Data representation of a BIM model. The first solution provided by this research project extracts and represents the 2D building footprint geometry of a 3D BIM building. This facilitates the exploration of the building in 2D GIS applications. The second solution demonstrates a method of representing a BIM building as 3D GIS, allowing for the representation of a BIM building in a 3D GIS context. The third solution in the solutions toolkit provides a more accessible Linked Data BIM building representation, where an ifcOWL building is represented using the Building Topology Ontology (BOT). This is in an attempt to simplify a BIM building and provide a more user friendly method of interlinking the data with other Linked Data domains. The fourth and final solution implemented in this project performs GeoSPARQL querying of the processed building data. The motivation behind this solution is to create a method to align a building geometry with a corresponding building geometry defined in a different domain, interlinking the domains.

The results achieved by this project show that it is possible to integrate BIM and GIS through building geometry representation methods using Linked Data technologies. The extracted GIS geometries and representations closely resemble their parent BIM models. The solutions toolkit does not provide an answer for BIM and GIS integration, however, it does provides an initial exploration into the use of building geometries to connect the domains. This could open the door to new alternative methods of BIM and GIS integration being discovered.

Abstract

Geographical Information System (GIS) data describing a building can help enrich our understanding of a building and its surroundings. There currently exists challenges in relating a GIS building to its corresponding Building Information Modelling (BIM) building. This is primarily due to the use of different geometric representations and coordinate systems which act as a blocker in the integration of the two domains. Seamless movement between BIM and GIS would allow for the exploration of a how a BIM building interacts with its external environment.

This dissertation presents work that addresses the challenges in aligning BIM and GIS building representations by using a Linked Data approach. Industry Foundation Class (IFC) BIM models are processed as ifcOWL to extract a 2D GIS representation of the 3D BIM model, based on the geolocation of the building. GeoSPARQL querying can be performed on the data to extract 2D GIS geometries. This facilitates the interlinking of data with other domains, using the extracted 2D building geometry as an alignment property.

The complexities associated with developing BIM data act as a deterrent for new BIM developers. This dissertation explores the Building Topology Ontology (BOT) as a more accessible means of representing BIM data. This is an important step towards the iterative integration of ever more complex BIM models into the wider web of data.

Lastly, a method of representing a BIM model as 3D GIS is presented in this dissertation. A 3D GIS building is created from a 3D BIM model, facilitating the exploration of a BIM building in a 3D GIS context. In total, four BIM and GIS integration methods are presented in this document. Each method provides a different approach of relating BIM and GIS buildings, based on the varying needs of a user.

Acknowledgements

I would like to take this opportunity to thank and number of people who have helped make this dissertation possible.

Firstly, to my project supervisors, Prof. Declan O'Sullivan and Dr. Kris McGlinn. Their weekly meetings were a great source of motivation and guidance. The support they provided throughout the duration of this research project provided me with valuable insights and truly made the project enjoyable.

Secondly, I would like to thank my family and girlfriend Ellen. Their continual words of encouragement during this research project, and throughout my time at Trinity College, helped me strive to achieve my full potential. For this I am sincerely grateful.

Table of Contents

1. Introduction 1
1.1 Problem Area 1
1.2 Research Objectives 1
1.3 The Solution Toolkit 2
1.4 Report Structure and Contents 3
2. State of the Art 4
2.1 Geographic Information Systems (GIS) 4
2.2 Building Information Modelling (BIM) 5
2.3 Integrating BIM & GIS 5
2.3.1 Data Level Integration 6
2.3.2 Process Level Integration 6
2.3.3 Application Level Integration 7
2.4 Industry Foundation Class (IFC) 8
2.5 Linked Data 9
2.5.1 Web of Documents 9
2.5.2 Web of Data 9
2.5.3 Resource Description Framework (RDF) 10
2.6 Linked Data & BIM 10
2.6.1 ifcOWL 11
2.6.2 Building Topology Ontology 11
3. Design and Implementation 13
3.1 Solutions Toolkit Design 13
3.2 ifcOWL Building Footprint Representation as 2D WKT 14
3.2.1 Motivation 14
3.2.2 Algorithm Design 15
3.2.3 Algorithm Implementation 16
3.2.3.1 Building Geolocation Extraction 17
3.2.3.2 Building True North Orientation Extraction 18
3.2.3.3 Building Wall Extraction and Representation as WKT 18
3.3 ifcOWL Building Footprint Representation as 3D OBJ 21
3.3.1 Motivation 21
3.3.2 Algorithm Design 22
3.3.3 Algorithm Implementation 23
3.3.3.1 Extract Wall Position, Direction and Length 24
3.3.3.2 Determine Wall Height and Building Storey Level 24

3.3.3.3 Define OBJ Wall Vertices 26
3.3.3.4 Create OBJ Building Model 27
3.4 ifcOWL to Building Topology Ontology (BOT) 28
3.4.1 Motivation 28
3.4.2 Algorithm Design 28
3.4.3 Algorithm Implementation 30
3.4.3.1 Create BOT Building Component Definitions 30
3.4.3.2 Create BOT Building Component Relationship Hierarchy 31
3.5 GeoSPARQL Building Footprint Alignment 34
3.5.1 Motivation 34
3.5.2 Algorithm Design 35
3.5.3 Algorithm Implementation 36
3.5.3.1 Determine Total Area of WKT Geometry 36
3.5.3.2 Attach Area Value to Building Representation 37
3.5.3.3 GeoSPARQL Querying of the Processed Building Data 37
4. Evaluation 39
4.1 Results 39
4.1.1 ifcOWL Building Footprint Representation as 2D WKT Results 41
4.1.2 ifcOWL Building Footprint Representation as 3D OBJ Results 44
4.1.3 ifcOWL to Building Topology Ontology (BOT) Results 47
4.1.4 GeoSPARQL Building Footprint Alignment Results 48
4.2 Evaluation of the Solutions Toolkit 49
4.2.1 Evaluation of ifcOWL Building Footprint Representation as 2D WKT 50
4.2.1.1 Accuracy and Usability of Results 50
4.2.1.2 Runtime Performance of the Solution 52
4.2.1.3 Limitations 53
4.2.2 Evaluation of ifcOWL Building Footprint Representation as 3D OBJ 53
4.2.2.1 Accuracy and Usability of Results 53
4.2.2.2 Limitations 55
4.2.3 Evaluation of ifcOWL to Building Topology Ontology (BOT) 55
4.2.3.1 Usability of Results 56
4.2.3.2 Limitations 57
4.2.4 Evaluation of GeoSPARQL Building Footprint Alignment 57
4.2.4.1 Accuracy of Results 57
4.2.4.2 Limitations 59
4.3 Project Challenges 59
4.3.1 Learning New Technologies 60
4.3.2 Shortage and Consistency of Test Data 60
4.3.3 Geometry Skewing Issues 61

5. Conclusion 63
5.1 Future Work 64
5.2 LDAC Research Paper Submission 65
5.3 Concluding Remarks 65

1. Introduction

1.1 Problem Area

Effective management of our smart buildings and cities requires building data that is structured, accessible and reliable [1]. Building Information Modelling (BIM) is the primary method of providing such building data, representing the physical and functional characteristics of a building in a 3D model. It is difficult, however, to describe BIM models in their overall geospatial context. This could allow for important external environmental information to be overlooked. The combination of BIM and Linked Data (LD) [2] has the potential to meet the requirements for storing, sharing and interlinking BIM with other data sources. BIM represented as Resource Description Framework (RDF) [6] would facilitate the interlinking of BIM data with other Linked Data resources on the web [3].

Geographical Information System (GIS) data describing a building can help enrich our understanding of a building and its surroundings. The use of different geometric representations and coordinate systems in GIS and BIM leads to difficulties in moving between these representations. There currently exists a disconnect between the two representations, making it a challenge to a relate a 2D GIS building with its corresponding 3D BIM model. The construction industry uses Industry Foundation Classes (IFC) [5] as the standard method to define a BIM model. In IFC, buildings and their components are modelled as objects with relationships and hierarchies between objects defined. Research around the representation of geospatial data in combination with IFC is ongoing [4], but it remains the case that there are still issues related to how IFC models its geospatial data. There is no standard process in place to convert a Cartesian coordinate IFC entity to GIS. New methods of carrying out this conversion process are required until an agreement on coordinate systems is established between the differing domains.

1.2 Research Objectives

The goal of this research project is to bridge the gap between GIS and BIM building representations through a Linked Data approach. This involves the extraction of geometries from 3D BIM models in an attempt to relate a GIS building with its corresponding BIM model. The disconnect between the two representations acts as a blocker in relating a building to its wider geospatial context. This can hinder the exploration of how a building interacts with its

surroundings such as infrastructure (roads, bridges) and natural features (rivers, lakes, mountains).

This research project examines the ifcOWL [7] representation of a BIM model. The primary objective is to derive a process that extracts a 2D geometry from an ifcOWL representation. The extracted geometry is reattached to the ifcOWL building representation as a GeoSPARQL geometry [8]. Additionally, a Building Topology Ontology (BOT) [9] representation of the building is created with the extracted geometry attached. Querying of the enriched ifcOWL and BOT building representations facilitates the interlinking with other Linked Data domains.

The extracted building geometry is represented as both 2D Well-known text (WKT) [10] and 3D Wavefront OBJ [11]. The objective here is to have multiple methods by which a building can be represented, outside of its BIM context. This increases the possible use cases of the extracted building geometry. The WKT geometry attached to the ifcOWL and BOT representations serves the use of examining the building in a 2D GIS context. The OBJ building representation has potential for use in computer graphics and 3D GIS applications.

The final objective of this project is to explore methods by which GeoSPARQL functions can be used to align two building geometries. GeoSPARQL offers topological relations which allow for the discovery of relationships between geometries. The processed ifcOWL and BOT building representations are queried to align their geometries with the same building geometry sourced from a different dataset. This facilitates the interlinking of the building with other Linked Data domains.

1.3 The Solutions Toolkit

The solutions presented in this research project all work towards the goal of connecting GIS building geometries with their 3D BIM models. Each solution provides a different method of contributing to this goal. A solution can be seen as being a member of a wider "solutions toolkit". The toolkit is the set of all solutions presented in this research project. The following solutions comprise the solutions toolkit:

1. A method to extract the building geometry from ifcOWL, convert to WKT and reattach to the ifcOWL representation.
2. A method to create a Building Topology Ontology (BOT) representation of the ifcOWL building with associated WKT geometry.
3. A method to convert the extracted building geometry to OBJ.
4. A method of building geometry alignment with GeoSPARQL querying.

All solutions are discussed in Section 3 of this document.

1.4 Report Structure and Contents

This document is structured as follows.

Section 2 examines the state of the art. A discussion on Geographical Information Systems (GIS) and Building Information Modelling (BIM) is provided. IFC, Linked Data and BOT are also explored, with reference to previous work carried out in the field.

Section 3 discusses the design and implementation of the project solutions. The process by which a geometry is extracted from an ifcOWL building is presented along with the method of conversion to Well-known text (WKT) and OBJ. This section also presents the method of GeoSPARQL building alignment.

Section 4 provides an evaluation of the solutions presented, giving images of the resulting geometries. A discussion on the accuracy and usability of the results is also given.

Section 5 discusses the conclusions reached in this research project and how the solutions provided contribute to the problem area. A discussion on future work is also given.

2. State of the Art

This chapter discusses the state of the art of the technologies and concepts explored in this project. The state of the art of Geographic Information Systems (GIS) and Building Information Modelling (BIM) is first examined in Sections 2.1 and 2.2, discussing geometry representations, uses cases and limitations for each. An evaluation of research around the integration of BIM and GIS is provided in Section 2.3, primarily focusing on data level, process level and application level integration of BIM and GIS. The current BIM representation standard, Industry Foundation Classes (IFC), is next discussed in Section 2.4, detailing the difficulties in working with the IFC schema. This project uses a Linked Data approach to BIM and GIS integration, section 2.5 explores the state of the art in Linked Data technologies. Following this is Section 2.6, which examines current Linked Data BIM standards, ifcOWL and the Building Topology Ontology (BOT), concluding the state of the art chapter of this document.

2.1 Geographic Information Systems (GIS)

Geographic Information Systems (GIS) are information systems with an added geo-reference. The geospatial information associated with GIS systems allow for spatial analysis to be performed on the system. GIS data has a vast array of use cases including the monitoring of weather systems across a region, predicting population densities in a city and visualising real time traffic jams in a certain location. Analysis of GIS data gives important location based insights which may have previously been overlooked.

Geospatial information in a GIS system typically include the coordinates of features of an object, based on the real world location (geolocation) of the object. The relationship between features of the object can also be defined. Such features may include the walls of a building, and how certain walls relate to one another by being associated to a room within the building. GIS features are commonly represented in a 2D coordinate system. A building would be represented by its top-down building footprint. A GIS system representation of a building will also include the building within the context of other geospatial features, such as infrastructure and natural features. This makes the information relevant to both building designers and public planning organisations, planning for navigation, fire services, energy grids, etc.

Recent research in GIS primarily focuses on the development of 3D GIS [13]. Such systems facilitate the modelling of complex internal structures of buildings, tunnels and bridges in the

geospatial domain. City Geography Markup Language (CityGML) is currently the state of the art standard in 3D GIS modelling, providing an XML based data model for the storage and exchange of 3D models [14]. This 3D GIS modelling approach is closer to Building Information Modelling (BIM) than 2D GIS, however work still remains in providing seamless interaction between the two representations.

2.2 Building Information Modelling (BIM)

The concept of Building Information Modelling (BIM) has been created to support the vast amount of data associated with buildings. This data is generated across the building's life cycle (BLC) and requires maintenance and management throughout the BLC. BIM describes an integrated data model for storing all information relevant to the BLC, typically relating to the functional and physical characteristics of a building [15]. This primarily includes a 3D model of the architectural design, detailing the positions and dimensions of a building's walls, rooms, windows, doors, roof, etc. BIM also facilitates the inclusion of non-physical building features such as the building costs, accessibility, safety, security and sustainability [16]. BIM is therefore capable of capturing all aspects of a building that exist throughout the BLC, aiding stakeholders at all stages of the BLC. As the authors state in [13], "it is clear that BIM is not just a piece of software, but also a process that contributes to the workflow and project delivery process."

However, a BIM model lives in isolation, it is oblivious to the external world that surrounds the building. This is a limitation of BIM. The lack of geospatial analysis functionality available from a BIM model restricts the knowledge one may have of a building's surrounding environment. The impact a building may have on its environment could be overlooked if the appropriate geospatial analysis is not performed.

2.3 Integrating BIM & GIS

There is potential to further improve existing use cases by integrating Building Information Modelling (BIM) with Geographic Information Systems (GIS). This integration would enrich BIM datasets, allowing stakeholders to observe new building features that may not have been visible using BIM alone, in particular relating to the building's wider geospatial context. However, the different geometry representations and coordinate systems used by BIM and GIS make it challenging to interlink the two data types. There is no standard process currently in place to transform BIM to GIS. Recent efforts to integrate BIM and GIS can be observed at three levels: data level, process level and application level [13, 17].

2.3.1 Data Level Integration

The development of new data modelling standards is one area of research that works towards the integration of BIM and GIS. LandXML, and subsequently InfraGML [26], was developed by the Open Geospatial Consortium (OGC) Land And Infrastructure Domain Working Group (LandInfraDWG) to perform such data level integration [18]. InfraGML models civil engineering infrastructure and the development of land and terrain. IndoorGML, another GIS data modelling standard, models the indoor topology of buildings, closely aligning with BIM models [27]. An IndoorGML editor has been built to help develop IndoorGML models [19]. This editor can support both BIM data and geospatial information.

Other methods of BIM and GIS data level integration focus on the conversion and transformation of existing standards, primarily focusing on the conversion of BIM and GIS geometries. This method of BIM and GIS integration relates closely to the geometry conversion methods used in this project. Research has been carried out to integrate BIM Industry Foundation Classes (IFC) with CityGML using geometry conversion. Geiger, et al. [20] explored how to make both IFC and CityGML compatible in one system by simplifying the complexity of the IFC model geometry. Another data level integration method, the Feature Manipulation Engine (FME) [28], applies an Extract Transform Load (ETL) process to integrate BIM and GIS data. Data is extracted from both BIM and GIS sources and loaded into a data warehouse in a common format. The ETL method used by the FME supports the bidirectional reading and writing of IFC and CityGML data [21].

2.3.2 Process Level Integration

The use of semantic web technologies provide potential solutions to integrate BIM and GIS at a process level [13]. Semantic web technologies exist to connect related data on the web, to "allow data to be shared effectively by wider communities, and to be processed automatically by tools as well as manually" [22]. The goal of connecting related data on the web makes semantic web technologies an ideal candidate to perform the integration of BIM and GIS data, connecting the two related domains. The process level use of semantic web technologies to integrate BIM and GIS aligns closely with the solutions implemented in this project.

Semantic web based reference ontologies can be used to store and share the differences between BIM and GIS. McGlinn, et al. [23] demonstrate a method of uplifting relational GIS building data into Resource Description Framework (RDF), the framework for describing resources on the semantic web. This is achieved by using the OSi ontology [29] and the ifcOWL ontology to redefine the data using BIM IFC standards. The data is then ready to be interlinked with other building domains. The work carried out in [23] is built on in [24], where the geolocation of an ifcOWL

building is extracted as a latitude and longitude coordinate. This coordinate is the real world location of the IFC building in a GIS context, connecting a BIM building with the location of its corresponding GIS building.

2.3.3 Application Level Integration

Integration of BIM and GIS at an application level does not change the source data and a reference ontology is not created. Application level integration usually refers to a plugin being installed in BIM software that extracts the GIS information from the building [13]. This GIS information is then stored in a database which is made accessible to GIS software. Application level integration of BIM and GIS is useful for special cases and do not scale. This is due to the number of different BIM and GIS software applications available, requiring a plugin and shared database to be made for each use case.

A comparison of BIM and GIS integration methods is given in Table 2.1 (adapted from a paper by Liu, et al. [13]). Integration methods are rated on effectiveness, extensibility, effort and flexibility (EEEF) on a *case by case*, *low*, *medium* and *high* scale. Semantic web technologies are rated as *high* for effectiveness, extensibility and effort, with a rating of *medium* for flexibility. This gives semantic web technologies the best overall EEEF rating when compared to the other integration methods considered, even though the effort required is high. These ratings support the use of semantic web technologies in this project to integrate BIM and GIS.

Integration Methods	**Effectiveness**	**Extensibility**	**Effort**	**Flexibility**
New standards and models	case by case	case by case	case by case	case by case
Conversion, translation and extension of existing standards (manual)	medium	high	high	medium
Conversion, translation and extension of existing standards (semi-automatic)	medium	medium	medium	medium
Semantic web technologies	high	high	high	medium
Services-based methods	high	low	high	low
Application focused methods	case by case	low	low	low

Table 2.1: *"Comparison of integration solutions by "EEEF" criteria." [13]*

Alternative BIM and GIS integration approaches to the ones discussed above do exist, although they tend to cater for more specific use cases. Karan et al. [35] explore the use of remotely sensed light detection and ranging (LiDAR) data to generate 3D GIS topography models of construction sites. Irizarry et al. [36] integrate BIM and GIS by providing a solution to find the optimal location

of a construction crane tower on a construction site. A 3D model of the crane is visualized in a 3D virtual world alongside BIM building models, allowing construction site managers to pinpoint the most efficient crane location. These are unique approaches that are not entirely suitable for generic BIM and GIS building integration at scale.

It is clear that a number of methods and tools exist to integrate BIM and GIS. However, a single integration standard is yet to be defined. Table 2.1 above tells us that semantic web technologies have the highest potential in delivering such a standard, although the required effort is high. The disparities between BIM and GIS building representations require detailed knowledge of each schema. Furthermore, semantic web BIM and GIS integration methods require the user to be familiar with Linked Data theory and technologies. This contributes to high effort needed and increases the barrier for entry to Linked Data BIM development. This project aims to provide accessible semantic web based BIM and GIS integration processes that focus on the building geometry representation, removing the high level of effort required by the user.

2.4 Industry Foundation Class (IFC)

Standardisation is an important part of ensuring data interoperability. Industry Foundation Classes (IFC), developed by buildingSMART [30], is the current leading standard for BIM in the Architecture Engineering and Construction (AEC) industry. In IFC, buildings and their components are modelled as objects. Relationships and hierarchies between objects are defined, giving a semantically rich definition of a building. IFC uses the EXPRESS data modelling language to define a building and its component hierarchy. IFC is also the only BIM currently an ISO PAS standard (ISO 2013), and so it remains the de facto standard for BIM in AEC. The AEC industry relies on IFC for modelling core building processes (architecture, structural analysis, control, etc.), enabling information to be passed between different stakeholders across the BLC. IFC has serializations in STEP, XML and RDF alongside an OWL version of IFC4, called ifcOWL [7].

IFC, however, has yet to make its expected impact across the AEC industry, primarily due to its complexity. The complexity arises from the extensive component hierarchy native to IFC. This component hierarchy can be challenging for new software developers who are unfamiliar with the IFC schema. A generic two storey building modelled in IFC can result in an IFC file thousands of lines long. Simple geometry extractions require prior knowledge of the IFC schema, which proves as a barrier to its use. The complexity involved in learning and developing IFC can lead to IFC developers being reluctant in publically releasing their IFC models for free. The lack of IFC examples available online further contribute to the barrier for entry to IFC development, in turn, hindering the progress in finding a solution to the seamless integration of BIM and GIS.

2.5 Linked Data

Linked Data (LD) is an approach to expose, share, and connect related data, which was not previously linked, on the Web [2]. LD makes use of standardized technologies to define and relate data, working towards an overall goal of having a web of interlinked data that is available for both human and machine consumption. The concept of Web of Documents vs Web of Data is discussed when comparing the current state of the Web with a Linked Data approach of the Web. Linked data works towards a Web of Data.

2.5.1 Web of Documents

The Web of Documents is designed for human consumption. Traditionally, the Web of Documents provides a textual format such as CSV, XML, HTML, etc. The lack of structure of these documents make it difficult for machine parsing. Linking documents is an added challenge, where the relationship between two documents is implicit, giving little semantics to machines. The Web of Data aims to solve the interoperability issues observed with the Web of Documents by providing a standardized structured approach to define and relate data on the web.

2.5.2 Web of Data

Linked Data (LD) works towards a Web of Data that is both human and machine consumable. Data is highly structured with explicit links being defined between related pieces of data. This gives better semantics to machines, allowing them to traverse the Web with greater ease. LD uses the following technologies to define and relate data: Uniform Resource Identifiers (URIs) to name data, Resource Description Framework (RDF) to represent and relate data and the HTTP infrastructure to retrieve data. The *Linked Data Principles*, outlined by Bizer, et al. [25], define the rules by which Linked Data should be published on the web.

Linked Data Principles:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful RDF information
4. Include links to other URIs so that they can discover more related things

These principles form a basis for how the Web of Data is organised. A human requesting a Linked Data resource is presented with the content in a human readable format such as a HTML web

page. A machine requesting the same resource is given a RDF version of the content. This ensures that both humans and machines gain the same knowledge from the same resource.

2.5.3 Resource Description Framework (RDF)

The Resource Description Framework (RDF) represents data using a directed graph model. An RDF graph represents objects and object relationships using *subject-predicate-object* triples. The subject and object are the things being described, where the predicate is the relationship between these two things. Each node in the graph is a subject or object. The arc between two nodes represents the predicate. Figure 2.1 shows an example RDF graph describing a person who has name, email and height properties and also is a student of Trinity College Dublin, which has an address property and a founded property.

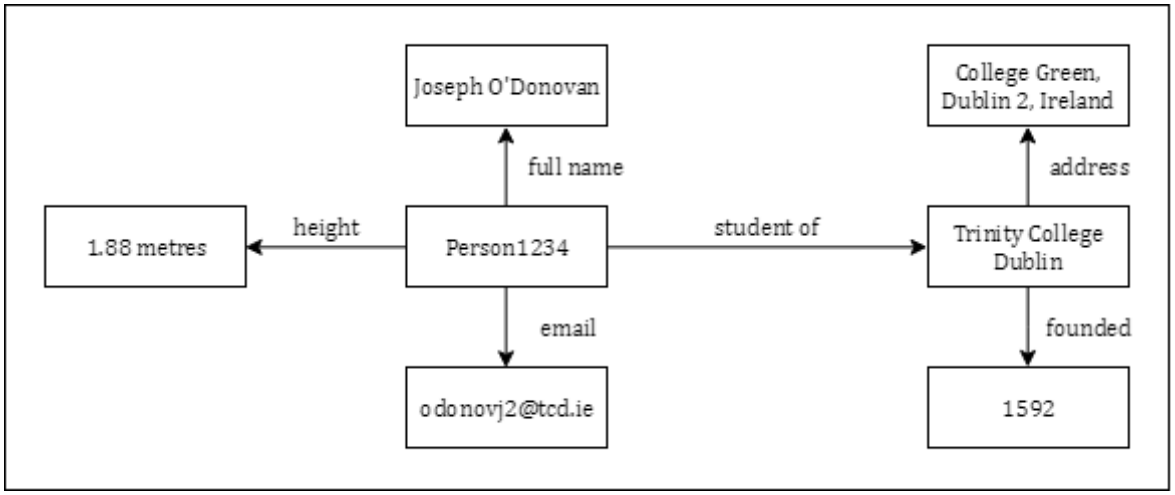


***Figure 2.1**: RDF graph example showing subject-predicate-object triples. The nodes of the graphs represent subjects and objects. The arcs are the relationships between the subjects and objects.*

2.6 Linked Data & BIM

Linked Data (LD) makes us of the Resource Description Format (RDF) to store and link data, allowing linking between domains across the web [2]. This interlinking of domains has significant potential in the AEC industry, where data related to different domains are generated and consumed across the building lifecycle (BLC). Each stage of the BLC, from design to construction and maintenance, require data sourced from a vast set of domains; building geometry and topology data, sensor data, behaviour data, geospatial data, etc. The combination of BIM and LD has the potential to meet the requirements for storing and sharing those data. However, those data have to be represented as or at least tagged using RDF.

The ifcOWL ontology and the Building Topology Ontology (BOT) provide BIM representations of a building using a Linked Data approach. Both ontologies serve as potential methods of BIM and GIS integration, interlinking the two domains.

2.6.1 ifcOWL

The ifcOWL ontology is the IFC Web Ontology Language (OWL) version of the IFC schema [7]. An official buildingSMART [31] standard, ifcOWL provides an RDF version of the IFC schema for use with Linked Data technologies. SPARQL querying of ifcOWL buildings is a current area of research, with example queries to meet specific use cases already being explored [52]. Querying of ifcOWL buildings opens the door for the interlinking of BIM with other domains, enriching the knowledge available about a building. An IFC building is transformed to ifcOWL using the EXPRESS data modelling language, capturing all building components and relationships given in the IFC schema. A direct mapping can be made between each RDF triple in an ifcOWL file to a corresponding building component in its original IFC file. Figure 2.2 shows an example ifcOWL RDF triple that relates `IfcBuildingStorey_479` and `IfcBuildingStorey_35065` to `IfcBuilding_434`, the building containing the two storeys.

```
inst:IfcRelAggregates_481
       ifcowl:globalId_IfcRoot      inst:IfcGloballyUniqueId_45211 ;
       ifcowl:ownerHistory_IfcRoot  inst:IfcOwnerHistory_12 ;
       ifcowl:relatingObject_IfcRelAggregates  inst:IfcBuilding_434 ;
       ifcowl:relatedObjects_IfcRelAggregates  inst:IfcBuildingStorey_479 ;
       ifcowl:relatedObjects_IfcRelAggregates  inst:IfcBuildingStorey_35065 .
```

Figure 2.2: *Example ifcOWL triples, relating two IfcBuildingStoreys to an IfcBuilding.*

2.6.2 Building Topology Ontology

The Building Topology Ontology (BOT) [9], currently under development within the auspices of the Linked Building Data on the Web community group, may provide a bridge between software developers and IFC. BOT provides descriptions for only the most fundamental properties of a building, in terms of its topology, in a Linked Data approach. The primary components of a building topology (walls, storeys, rooms, etc.) are modelled in BOT, avoiding the complexities observed in IFC. Additionally, BOT only permits a small finite set of relationships to be defined between BOT objects, further simplifying the building representation. The purpose of BOT is to support linking to other ontologies when details are required for other aspects of the building,

such as those related to geolocation, building products, automation and control, HVAC and energy as well as those data not traditionally within the scope of BIM, for example weather and energy tariffs. The simplicity of BOT makes the ontology more accessible when compared to the ifcOWL ontology, making it a more attractive medium to interlink BIM with other domains. Figure 2.3 gives an example BOT triple, relating the same `IfcBuilding_434` defined in Figure 2.2 above to its two building storeys, `IfcBuildingStorey_479` and `IfcBuildingStorey_35065`.

```
inst:IfcBuilding_434
        a               bot:Building ;
        geo:hasGeometry <urn:geom:pt:2hQBAVPOr5VxhS3Jl0O47h> ;
        bot:hasStorey   inst:IfcBuildingStorey_479 , inst:IfcBuildingStorey_35065.
```

Figure 2.3: *Example BOT triples, relating two IfcBuildingStoreys to an IfcBuilding.*

This project explores the use of both the ifcOWL and BOT ontologies to provide a solution to the integration of BIM and GIS. It is clear from comparing the ifcOWL triple definitions, in Figure 2.2, to the BOT triple definitions, in Figure 2.3, that the BOT triple definitions are more intuitive. The use of a non-complex solution such as BOT could aid in overcoming the difficulties associated with working with existing BIM standards such as IFC.

3. Design and Implementation

The Design and Implementation chapter of this report is divided into five sections. The first section, Solutions Toolkit Design, discusses the reasons behind choosing the four solutions implemented as part of the solutions toolkit. The following four sections describe the four project solutions, with each section giving the motivation, an overview of the algorithm design and a technical implementation for that solution.

3.1 Solutions Toolkit Design

This project implements four solutions that work towards the overall goal of integrating buildings defined in Building Information Modelling (BIM) and Geographic Information Systems (GIS). As discussed in Section 1.3 above, all four solutions comprise a "solutions toolkit," with each solution contributing to the project goal. This is to allow the user to select a BIM and GIS integration method that best suits their needs. Three of the four solutions focus on the extraction and representation of the building footprint geometry of an ifcOWL building. The fourth solution provides a method of building geometry alignment using GeoSPARQL. The four implemented solutions, listed below, were chosen to help solve different BIM and GIS integration issues. Each solution approaches BIM and GIS integration from a different viewpoint, with the primary focus being a GIS geometry representation of a BIM IFC building using a Linked Data approach.

The following are the four implemented solutions that comprise the solutions toolkit:

1. ifcOWL building footprint representation as 2D WKT
2. ifcOWL building footprint representation as 3D OBJ
3. ifcOWL building to Building Topology Ontology (BOT)
4. GeoSPARQL building footprint alignment

Initially, the project focused on integrating BIM and 2D GIS, with the main goal being the extraction of a 2D building footprint geometry from an ifcOWL building. This solution built on the work carried out by McGlinn et al. [24], where the 2D coordinate geolocation of an ifcOWL building is extracted. A process to extend the work in [24], to extract a 2D GIS building footprint geometry from an ifcOWL building, was chosen as the primary project solution. It was predicted

that this would have the greatest impact in 2D GIS, since 2D GIS applications typically represent buildings as top-down building footprints.

The project soon evolved into requiring an integration method for 3D GIS, where the ifcOWL building walls are represented as 3D OBJ for use in 3D GIS applications. This would allow the owner of a BIM model to view their building in a 3D GIS context, following a similar motivation to the ifcOWL building footprint as 2D WKT solution.

Ease of use is the primary motivation for the third solution, where the basic structural components of an ifcOWL building are represented using the Building Topology Ontology (BOT). BOT is a more straightforward ontology than ifcOWL, making it easier for user interaction. This solution still focuses on geometry representation, where the 2D WKT geometry is added to the BOT file, however it has the added benefit of BOT being more accessible for users.

The fourth and final solution implemented in this project works with the processed ifcOWL and BOT files from Solutions 1 and 3. GeoSPARQL querying of the processed building data enables building geometry alignment, facilitating the interlinking of Linked Data domains. This solution was chosen to help enrich the knowledge available on a Linked Data BIM model, by allowing the user to explore their BIM building in relation to other Linked Data domains. This solution completes the solutions toolkit proposed by this project and realizes the project goal, integrating a BIM building into GIS and, in doing so, interlinking the building with other domains.

3.2 ifcOWL Building Footprint Representation as 2D WKT

3.2.1 Motivation

A BIM building is represented as a 3D model, modelling the physical and functional characteristics of a building. Models vary in the level of detail depending on the use case. Comprehensive BIM models would include all structural building components as well as details such as the placement of furniture, electrical sockets and ventilation, along with non-physical building components such as the building costs and energy consumption. These finer details of a BIM model are not of great relevance to the context of this solution, where a focus is placed on the process to extract a 2D building footprint geometry from a 3D BIM model based on the building's wall structure.

The 2D geometry extracted from a 3D BIM model is the top-down view of the model, where the external walls of the building represent the building footprint. The motivation behind extracting the 2D building footprint geometry of a 3D BIM model is to define the BIM building in a 2D GIS context. This enables geospatial analysis to be performed on the BIM building, integrating the two domains. Furthermore, in a Linked Data context, the extracted geometry could be used to align

the building to its corresponding building defined in a different Linked Data dataset. The alignment of two building geometries therefore becomes a method of interlinking Linked Data domains. Extraction of a 2D building footprint geometry from a BIM model is a necessary step in working towards achieving the project aim.

3.2.2 Algorithm Design

The algorithm to extract and represent a 2D building footprint geometry from an ifcOWL building can be broken down into three steps:

1. Building geolocation extraction
2. Building true north orientation extraction
3. Building wall extraction and representation as WKT

The geolocation of a building is useful in determining the exact location of a building. It allows positioning of a building in the world coordinate system through use of a latitude and longitude coordinate. The geolocation of an ifcOWL building is extracted to determine the real world position of the building. The extracted geolocation also acts as a reference point in the positioning of a building walls. Extraction of the geolocation is the first step in connecting a BIM building to its corresponding GIS building.

The second step is to extract the building's true north orientation. This is the real world direction that the building is facing. The true north orientation angle is later used to rotate the building walls to the correct building orientation.

The final step in creating the building footprint is to extract and represent the walls of the building. Each wall in an ifcOWL building is defined by a position, direction and length, all represented as Cartesian coordinates in metres. These three wall properties, along with the building geolocation and true north orientation, facilitate the placement of a wall in its correct real world position. The walls are positioned relative to the building geolocation and then rotated to the true north orientation angle. Figure 3.1 shows a building footprint geometry where the walls, represented as red lines with white end points and transparent midpoint, are positioned relative to the geolocation (*Geo*), and rotated to the true north rotation angle (*TN*).

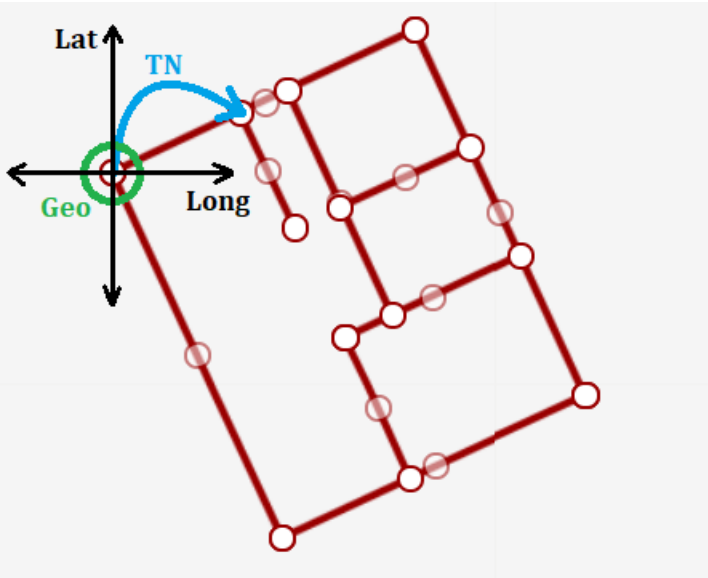


Figure 3.1: *A resulting 2D GIS building footprint geometry. The walls, represented by the red lines with white endpoints and transparent midpoint, are positioned relative to the geolocation coordinate (Geo). The walls are then rotated to the true north angle (TN) to face the real world building orientation.*

Each external building wall is transformed to its real world position and represented as a Well-known text (WKT) [10] geometry string. WKT is a string based geometry representation language used to represent 2D vector geometries on a map. This is ideal for representing the building geometry in a 2D GIS context. Additionally, GeoSPARQL supports geometry representation as WKT. This enables the use of GeoSPARQL topology relationship functions in SPARQL queries to interact with WKT geometries. The WKT geometry representing the 2D building footprint of an ifcOWL building is inserted back into the ifcOWL file as an RDF triple. This processed ifcOWL file is now ready for GeoSPARQL querying.

3.2.3 Algorithm Implementation

As mentioned in Section 3.2.2 above, the algorithm to extract and represent a 2D building footprint geometry from an ifcOWL building has the following steps:

1. Building geolocation extraction
2. Building true north orientation extraction
3. Building wall extraction and representation as WKT

The implementation of these steps is detailed below as three separate subsections, Sections 3.2.3.1, 3.2.3.2 and 3.2.3.3. This solution is implemented using a Java Apache Jena project.

3.2.3.1 Building Geolocation Extraction

The geolocation of a building is useful in determining the exact location of a building. It allows positioning of a building in the world coordinate system through use of a latitude and longitude coordinate. The geolocation of an ifcOWL building is extracted to determine the world position of the building and assists in the relative positioning of a building's walls. Walls have a local placement relative to the geolocation of the building.

Apache Jena supports interaction with RDF files and is used in this project to extract RDF triples from the ifcOWL building files. The `refLatitude_IfcSite` and `refLongitude_IfcSite` of an ifcOWL building represent the latitude and longitude values of the geolocation coordinate [37]. Recursive traversal of both attributes, using Apache Jena, extract their respective values. These values are given in the format of degrees, minutes, seconds and millionth of a second. Conversion to decimal degrees is necessary for generation of Well-known text (WKT) geometries. A WKT point is created with the decimal degrees latitude and longitude in the form `POINT(longitude latitude)`. A GeoSPARQL `hasGeometry` property is created for the `IfcSite`. This property relates to a GeoSPARQL `asWKT` property, containing the WKT point string that represents the building geolocation. These additions to the building model are subsequently present in the output file. Figure 3.2 gives an example ifcOWL snippet of an `IfcSite`, taken from an output ifcOWL file. The geolocation is defined as a GeoSPARQL geometry which is represented as a WKT point coordinate.

```
inst:IfcSite_389
      a                   ifcowl:IfcSite ;
      [More properties defined here] ;
      geo:hasGeometry     <urn:geom:pt:0KMpiAlnb52RgQuM1CwVfd> .


<urn:geom:pt:0KMpiAlnb52RgQuM1CwVfd>
      geo:asWKT     "POINT (8.436539 49.100435)"^^geo:wktLiteral .
```

Figure 3.2: *An example ifcOWL triple definition of the IfcSite with added geolocation. The geolocation is defined as a GeoSPARQL geometry. This geometry is then defined as a WKT point. A number of the IfcSite properties have been removed from this example for brevity.*

3.2.3.2 Building True North Orientation Extraction

A Building represented in the IFC format typically represents a real world building. This building has a real world orientation, or direction, that the building is facing. An ifcOWL representation of a building incorporates a building's orientation with the true north direction of the building. This is the direction of the true north relative to the underlying coordinate system, given by a direction within the xy-plane of the coordinate system. If not given, the true north defaults to the positive direction of the y-axis [38].

The `trueNorth_IfcGeometricRepresentationContext` defines the true north direction of an ifcOWL building as an xy-coordinate, where both x and y are greater than or equal to 0 and less than or equal to 1. The angle between this coordinate and the positive direction of the y-axis (0, 1) is calculated. The arc tangent of the positive distance between the true north x-coordinate and 0, and the negative distance between the true north y-coordinate and 1 gives the angle (in degrees) between the two coordinates. This angle is later used in the program to rotate wall coordinates to the true north orientation.

3.2.3.3 Building Wall Extraction and Representation as WKT

A process for extracting the walls of an IFC building is necessary to create a building footprint. The building footprint facilitates alignment of the 3D IFC geometry with a 2D GIS geometry. IFC geometry models can contain two wall types: `IfcWallStandardCase` and `IfcWall`. The former is used for occurrences of walls that have a non-changing thickness. The latter is used for occurrences of walls that have a changing thickness, or have a non-rectangular cross section [39]. A hierarchy of entity placements define where a wall is positioned in the building's global coordinate system. The IFC Site of a building sits at the top of the hierarchy. The IFC Building is then positioned relative to the IFC Site. Each IFC Building Storey is positioned relative to the IFC Building. All walls on a storey are placed relative to the IFC Building Storey, completing the hierarchy.

The following process describes the extraction of walls represented as an `IfcWallStandardCase`. An `IfcWallStandardCase` is defined by three high level attributes:

1. A position (local placement)
2. An orientation (direction)
3. A length (defined by a polyline magnitude)

The IfcLocalPlacement of a wall specifies the position of a wall relative to the IFC Building Storey it is located on. The position is defined as a xyz-coordinate, denoting how far the wall is positioned from the origin. This origin is the origin of the building storey that contains the wall. All test data used in this project had the origin of the IFC Building Storey as the default origin (0, 0, 0), which, in turn, was the same origin as the IFC Building and IFC Site origins.

The IfcLocalPlacement of a wall also specifies the wall direction. The direction is defined by a xyz-coordinate. The default value for the direction coordinate is (0, 0, 0), indicating a positive rotation of 90 degrees around the z-axis. A value of 1 or -1 for the x-coordinate indicates a positive rotation around the z-axis of 270 or 90 degrees respectively. A value of 1 or -1 for the y-coordinate indicates a positive rotation around the z-axis of 0 or 180 degrees respectively.

The length of an IfcWallStandardCase is specified by an IfcProductDefinitionShape. This shape defines the IfcPolyline points that represent the length of the wall. A polyline is constructed by two xy-coordinate points, the xy-origin (0, 0) and a length along the positive y-axis (0, *Ylength*), where *Ylength* is the length of the wall. The two polyline coordinates are extracted for each wall of the building.

Positioning of the wall in the global place requires three steps:

1. Wall direction rotation
2. Wall global placement
3. True north rotation

The two polyline coordinates of each wall are first rotated to the IfcLocalPlacement direction. These, now rotated coordinates, are translated to the local position of the wall given by the IfcLocalPlacement position. As stated above, the hierarchy of xyz local placements for each of the IFC Site, IFC Building and IFC Building Storey are positioned at the global origin. This infers that the positioning of a wall, based on its local placement, translates the wall to its correct global position. The final step in global wall placement is the rotation of a wall to its true north orientation. The true north angle of rotation (determined in Section 3.2.3.2 above) is used to rotate wall coordinates to their correct global orientation. This three step procedure of global wall positioning is executed for each wall extracted from the ifcOWL file.

Figure 3.3 below shows an example transformation for a wall with the following attributes:

1. IfcLocalPlacement direction = (-1, 0, 0)
2. IfcLocalPlacement position = (10, 12, 0)
3. IfcPolyline points (0, 0) to (0, 10)

The wall in Figure 3.3 has an initial polyline length of 10 metres. The direction value has -1 for the x-coordinate, indicating a positive rotation of 90 degrees around the z-axis. The global position of the wall is a distance of 10 metres away from the geolocation (origin) on the x-axis and a distance of 12 metres away from the geolocation (origin) on the y-axis. The image on the left in Figure 3.3 shows the line after it has been rotated as per the wall direction. The image on the right shows the wall after it has been translated to its correct global position within the building. The true north rotation is not shown in Figure 3.3. The wall coordinates in the XYZ coordinate system start as (0, 0, 0) and (0, 10, 0). After rotation of 90 degrees the wall has the XYZ coordinates (0, 0, 0) and (-10, 0, 0). After translation, using the wall positioning coordinate of (10, 12, 0), the wall has the XYZ coordinates (0, 12, 0) and (10, 12, 0), as can be seen in the image on the right in Figure 3.3.

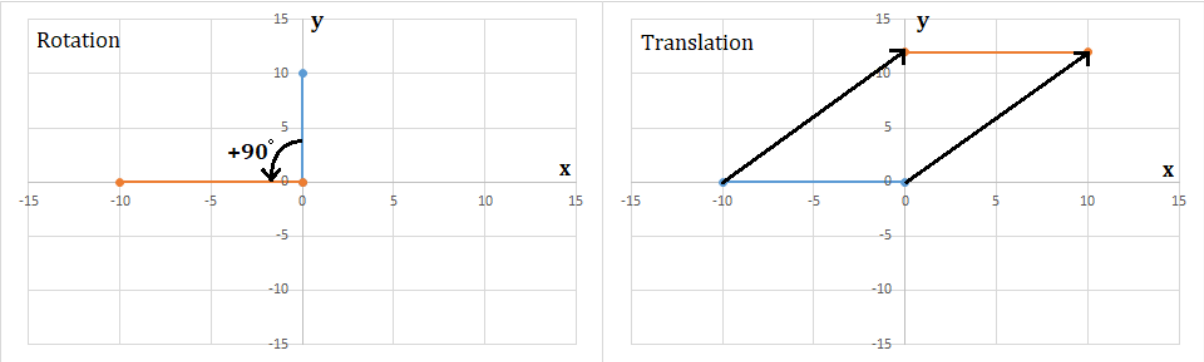


Figure 3.3: *Example rotation and then translation of a wall with initial polyline coordinates of (0, 0) and (0, 10). The orange line in the image on the left is the resulting rotated line. The orange line in the image on the right is the resulting translated line, translating the previously rotated line via the wall position coordinates. Shown in both images is the x and y axes. The z-axis is represented such that it is coming out of the page.*

Wall coordinates, in the global coordinate space, are next rotated as per the true north angle and converted to a latitude and longitude decimal degrees value. The conversion process uses the extracted decimal degrees geolocation (determined in Section 3.2.3.1 above) as a point of reference. The decimal degrees geolocation is transformed to an xy-coordinate using the WGS 84 Web Mercator (EPSG:3857) projection [47], giving the geolocation as a coordinate in metres. The xy-coordinates of each wall, given in metres, are added to the metres geolocation coordinate. The resulting xy-coordinates for each wall are converted back to decimal degrees using the WGS 84 (EPSG:4326) projection [48], giving the wall's global latitude and longitude in decimal degrees. A WKT `LINESTRING()` is created for each wall which is then added to the IFC model of the building and subsequently added to the output file. The latitude and longitude of all walls are

accumulated throughout the program and are concatenated at the end to create a WKT `MULTILINESTRING()`. This WKT geometry is added to the IFC model of the building and subsequently added to the output file. Figure 3.4 below shows pseudocode that gives an overview of the entire wall extraction process.

```
// Extract site data
buildingGeolocationDegrees = extractGeolocation(IfcSite)
buildingGeolocationXY = convertToEPSG3857(buildingGeolocationDegrees)
trueNorthAngle = extractTrueNorthAngle(IfcSite)

// Extract wall data
wallPosition = extractPosition(IfcLocalPlacement)
wallDirection = extractDirection(IfcLocalPlacement)
wallLengthCoords = extractWallPolylineCoords(IfcProductDefinitionShape)

// Transform wall coords into global space
rotatedWallCoords = rotateCoords(wallLengthCoords, wallDirection)
translatedWallCoords = rotatedWallCoords + wallPosition
trueNorthWallCoords = rotateCoords(translatedWallCoords, trueNorthAngle)
globalWallCoordsXY = buildingGeolocationXY + trueNorthWallCoords

// Convert wall coords to lat/long decimal degrees & create WKT
latLongWallCoords = convertToEPSG4326(globalWallCoordsXY)
wktMultilineString = createWKTString(latLongWallCoords)
```

Figure 3.4: *Pseudocode giving an overview of the entire wall extraction and transformation process of an ifcOWL building.*

3.3 ifcOWL Building Footprint Representation as 3D OBJ

3.3.1 Motivation

Section 3.2 above discusses a process to extract a 2D building footprint geometry from an ifcOWL file and represent it as WKT. This results in a 2D geometry of the building that can be used with 2D GIS applications. However, the 2D geometry could not be used in a 3D GIS system.

Applications such as ArcGIS Pro [34] support the representation of 3D building models. This facilitates the geospatial analysis of a 3D buildings. 3D building models can be created in 3D modelling software and imported into ArcGIS Pro, which supports the importation of 3D models as 3D Studio Max (*.3ds), VRML (*.wrl), OpenFlight (*.flt), COLLADA (*.dae) and Wavefront OBJ (*.obj).

One of the solutions implemented in this project as part of the solutions toolkit extracts the wall structure of an ifcOWL building for representation as 3D Wavefront OBJ [11]. The motivation for

this solution is to provide a process to connect a 3D BIM model to its corresponding 3D GIS model. This would facilitate the geospatial analysis of a 3D BIM model in a 3D GIS system, allowing for owners of BIM models to explore their building model in a 3D GIS context. A standard process to integrate 3D BIM and 3D GIS works towards the overall goal of integrating BIM and GIS buildings.

3.3.2 Algorithm Design

OBJ uses a Cartesian xyz-coordinate system to represent a model. The algorithm to create a 3D OBJ representation of an ifcOWL building therefore needs to provide a z-axis value for each wall. This z-axis value is determined by a combination of the wall height and wall building storey level. Furthermore, OBJ defines vertices as metre values, using (0, 0, 0) as the origin. The algorithm to represent an ifcOWL building as 3D OBJ performs all wall transformations in metres, using the (0, 0, 0) origin as the reference point. This differs to a 2D GIS geometry, where a latitude and longitude geolocation acts as reference point in the positioning of walls.

The algorithm to represent an ifcOWL building as 3D OBJ includes the following steps:

1. Extract wall position, direction and length
2. Determine wall height and building storey level
3. Define OBJ wall vertices
4. Create OBJ building model

The position, direction and length of each wall is extracted using the same method as in Section 3.2. above. The wall length is first rotated to its defined direction and then translated to the correct global position using the (0, 0, 0) origin as the reference point. This is now a 2D representation of the ifcOWL building. A notion of wall height needs to be introduced for the 3D OBJ model.

The bounding box height of an ifcOWL wall is extracted as the value for the height of the wall. This is a value in metres. The global height of a wall in an OBJ model is a combination of the wall height and the building storey height. Each wall sits on a building storey level within the building. The building storey level for each wall is determined, with this value defining the z-axis placement of the bottom of the wall in the OBJ model.

A wall represented in 2D WKT is defined by a line-string with two points, the start coordinate and end coordinate of the wall, where the distance between the start and end coordinates is the length of the wall. This definition of a wall is not suitable for 3D OBJ representation, where a wall is defined as a plane and all wall planes positioned together represent the building. A wall plane is

bounded by four points, representing the bottom left, bottom right, top left and top right corner points of the wall. Each ifcOWL wall therefore corresponds to four OBJ XYZ vertices. The bottom left and bottom right corner vertices use the building storey level as their z-axis values whereas the top left and top right corner vertices use an addition of the building storey level and the wall height as their z-axis values. Figure 3.5 shows how the combination of the wall height and building storey level determine the z-axis placement of a wall. The image shows that the z-axis values of the top two vertices of the green wall are determined by adding the 1st floor building level (2 metres) to the wall height (2 metres), in this case totalling 4 metres. The bottom two vertices of the wall have z-axis values of the 1st floor building level, in this case 2 metres.

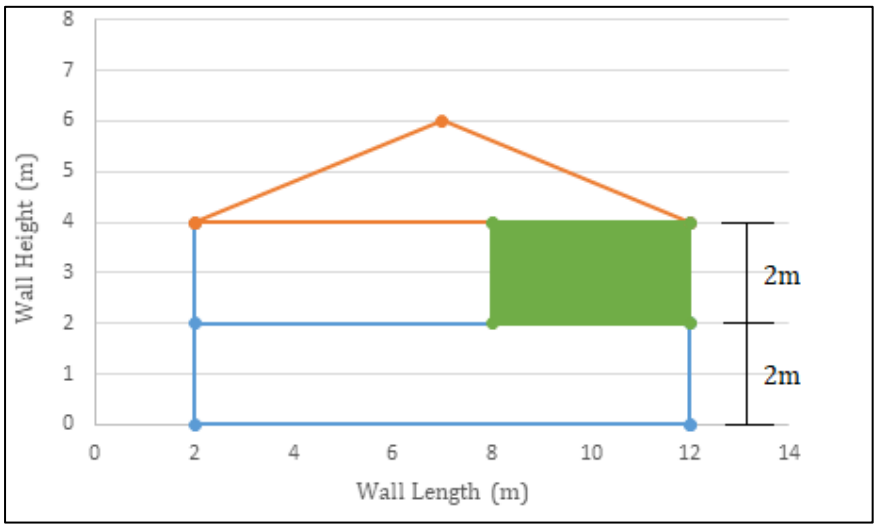


Figure 3.5: *A wall (green rectangle) of length 4 metres and height 2 metres is placed on the 1st floor building storey. The bottom vertices of the wall are placed at the level of the 1st floor, 2 metres up from the ground floor level. The top two wall vertices are positioned at a level which is the addition of the 1st floor level and the wall height, 4 metres up from the ground floor level.*

A 3D OBJ building model is defined by a description of the XYZ model vertices and a description of the model faces. Four XYZ vertices and one surface face is added to the OBJ model for each ifcOWL wall. A combination of the vertices and face create a plane in OBJ. This process is carried out for all ifcOWL walls, creating a structural 3D OBJ representation of the building.

3.3.3 Algorithm Implementation

The creation of a 3D GIS representation from a 3D BIM model works towards the overall project goal of BIM and GIS integration. 3D GIS systems facilitate the importation of 3D OBJ building models for exploration of the building in a 3D context. Creating a 3D GIS model of an ifcOWL

building has added complexities when compared to the creation of a 2D GIS model, primarily due to the added representation of wall height. The steps to create a 3D OBJ model of an ifcOWL building are defined in Section 3.3.2 above and are as follows:

1. Extract wall position, direction and length
2. Determine wall height and building storey level
3. Define OBJ wall vertices
4. Create OBJ building model

The implementation of these four steps are discussed in Sections 3.3.3.1, 3.3.3.2, 3.3.3.3 and 3.3.3.4 below.

3.3.3.1 Extract Wall Position, Direction and Length

The wall layout of a 3D OBJ building model is defined by the building's 2D geometry. The process to define the wall layout is identical to the algorithm implemented in Section 3.2 above, where a 2D WKT geometry is extracted from an ifcOWL building representation. However, in the case of 3D OBJ, all wall transformations are performed in a Cartesian xy-coordinate system in metres, using (0, 0, 0) as the origin. Resulting from this process is a 2D layout of the building walls using xy-coordinates. The concept of wall height needs to be introduced in order to create the 3D building model.

3.3.3.2 Determine Wall Height and Building Storey Level

Section 3.3.3.1 above creates a wall layout in the 2D xy-coordinate system. A 3D OBJ model defines objects using vertices in a xyz-coordinate system. A z-axis value is therefore needed for each wall. The z-axis value is determined by a combination of the wall height and the wall's building storey level.

The determination of wall height depends on the ifcOWL ontology version used to define the ifcOWL building. The test data in this project saw buildings that were defined using two ifcOWL versions, the ifcOWL `IFC4_ADD1` ontology [40] and the ifcOWL `IFC2X3_TC1` ontology. Different ontology versions are created depending on the underlying IFC schema versions, `IFC Version 4` [41] and `IFC2x Edition 3` [41]. IFC4_ADD1 is a newer version of the ifcOWL ontology, although only one test building used in this project was defined using this version

(AC20-FZK-Haus.ttl uses IFC4_ADD1, it can be downloaded from the online IFC repository at [43]).

IFC4_ADD1 Wall Height Extraction

An ifcOWL building using the IFC4_ADD1 ontology provides a definition of the bounding box dimensions of a wall via the IfcBoundingBox property. This bounding box property is extracted through the traversal of the IfcProductDefinitionShape, a property of each ifcOWL wall. The bounding box of a wall gives length dimensions in the x, y and z axes. The z-axis value is defined as a zDim_IfcBoundingBox length in metres. This z-axis value is extracted and used to represent the height of the wall for an ifcOWL IFC4_ADD1 building. The full relationship hierarchy to extract wall height as a zDim_IfcBoundingBox value for the file AC20-FZK-Haus.ttl can be seen in Appendix A1, where a value of 3.5 metres is extracted.

IFC2x2_TC1 Wall Height Extraction

Extraction of the wall height for an ifcOWL IFC2x3_TC1 building is more complex than its IFC4_ADD1 version counterpart. There are more steps required in determining the wall height and these steps differ from building to building. One method to determine the wall height of an IFC2x3_TC1 building is to extract the depth of a wall's IfcExtrudedAreaSolid property. The IfcExtrudedAreaSolid property of a wall can be discovered by iterating through the properties associated with a wall's IfcProductDefinitionShape. The depth value is defined by the depth_IfcExtrudedAreaSolid property of an IfcExtrudedAreaSolid resource. This is a value defined in metres and translates to the length of the wall along the z-axis. The full relationship hierarchy to extract wall height as a depth_IfcExtrudedAreaSolid value for the file smallhouse_saref.ttl can be seen in Appendix A2, where a wall height value of 4000 is extracted (this is a value in millimetres and not metres).

Another method of depth_IfcExtrudedAreaSolid wall height extraction from an ifcOWL IFC2x3_TC1 building uses a different relationship hierarchy as defined above. The method used depends on the how the ifcOWL IFC2x3_TC1 building is represented, with some building representations being more comprehensive than others. This alternative method uses an intermediary step where the IfcBooleanClippingResult property is used as a traversal property before reaching the IfcExtrudedAreaSolid resource definition. The height of the wall is ultimately determined as a depth_IfcExtrudedAreaSolid value. The full relationship hierarchy to extract wall height for the file Simple3-

`storeytestbuilding.ttl` using this alternative method can be seen in Appendix A3, where a wall height value of 2170 is extracted (this is a value in millimetres and not metres).

Building Storey Level Height

A combination of the wall height and building storey level determine the overall z-axis positioning of a wall in a 3D OBJ model. The relative placement of an ifcOWL wall determines the building storey level height. Each wall has an `objectPlacement_IfcProduct` predicate that defines a wall's `IfcLocalPlacement`. The `IfcLocalPlacement` definition describes how a wall is positioned relative to other building components, such as the building storey, the building and the site of the building. Recursive traversal of the values associated with the `IfcLocalPlacement` extracts the xyz-coordinate placement of a wall relative to its building storey. The z-axis value represents the height of the building storey level. This z-axis value, in metres or millimetres, is used in combination with the wall height to determine the global placement of the wall in the 3D OBJ building model.

3.3.3.3 Define OBJ Wall Vertices

The OBJ wall vertices can now be created as a results of the extraction of the wall height and its building storey level. As discussed in Section 3.3.2 above, a wall is represented by a xyz-coordinate plane in a 3D OBJ building model. Four OBJ vertices are needed to define a wall plane, representing the top left, top right, bottom left and bottom right corner points of the wall. The x and y values for all corner points are determined by the 2D xy-points of the wall, as calculated in Section 3.3.3.1. The top left and bottom left corner points share xy-coordinate values and the top right and bottom right corner points share their xy-coordinate values. The z-axis values for the bottom and top corner points differ. The bottom corner points are assigned a z-axis value that is the building storey level height, whereas the top corner points are assigned a z-axis value that is a combination of the building storey level height and the wall height. Figure 3.2 above gives an example of the overall wall height determination for a wall positioned on the 1st floor of a building.

A surface face is created for each set of four OBJ vertices. The face defines how vertices are related. Wall vertices are ordered in a clockwise formation, enabling the face, or plane of the wall, to be defined sequentially. An OBJ building wall is therefore defined by four vertices and one face. This process of defining a wall's vertices and face structure is executed for all ifcOWL building walls, creating the resulting 3D OBJ building model. Figure 3.6 below shows an example of four OBJ vertices and their face definition, defining how the vertices are connected to one another. The OBJ

vertices are the first four vertices in the OBJ file. The 1, 2, 3, 4 ordering of the face defines the order by which the first four vertices are to be connected, creating the face (wall plane).

```
v 0.0 10.0 0.0
v 12.0 10.0 0.0
v 12.0 10.0 2.7
v 0.0 10.0 2.7

f 1 2 3 4
```

Figure 3.6: *Four OBJ vertices, that represent the corner points of a wall, along with their face definition, defining how the vertices are to be connected.*

3.3.3.4 Create OBJ Building Model

The methods defined in Sections 3.3.3.1, 3.3.3.2 and 3.3.3.3 above are executed for all ifcOWL building walls. This leads to a collection of OBJ vertices and faces that represent the corner points and plane of each wall in the building. All OBJ vertices are written to the output OBJ file in sequential order, followed by the face definitions. Successful writing of the OBJ output file completes the process of 3D OBJ representation of an ifcOWL building. The OBJ model can now be imported into 3D modelling software, such as Blender [44], for inspection. Figure 3.7 below shows an example 3D OBJ representation of an ifcOWL building's wall structure.

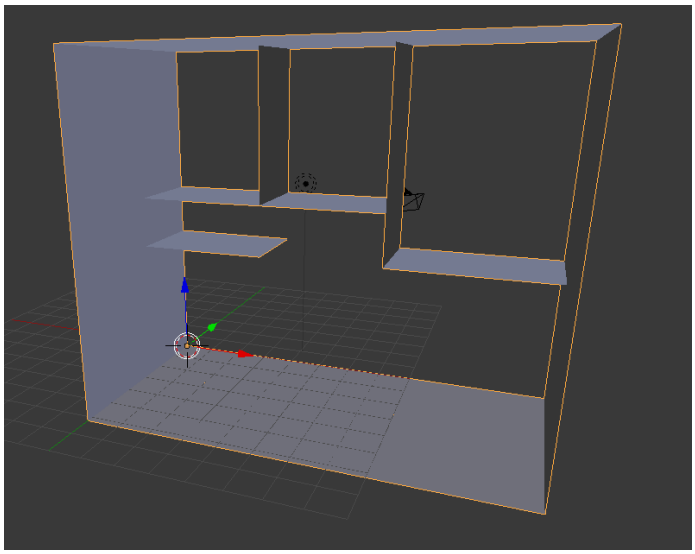


Figure 3.7: *Example 3D OBJ building model of an ifcOWL building. Both internal and external building walls are shown in this model.*

3.4 ifcOWL to Building Topology Ontology (BOT)

3.4.1 Motivation

The Building Topology Ontology (BOT) defines buildings as a topological structure. An emphasis is placed on the structural components of a building, excluding the complexities of ifcOWL. BOT does not provide as rich a definition as ifcOWL, however, the topological definitions it does provide are ideal for 2D GIS, where a building geometry is represented by the top-down building footprint. BOT is therefore a fitting ontology to model a 2D GIS topological view of an ifcOWL building. This is the motivation behind implementing an ifcOWL to BOT solution as part of the solutions toolkit.

The ifcOWL to BOT solution implemented in this project also serves the purpose of interlinking the building with other Linked Data domains. The 2D WKT geometry extracted from the ifcOWL building is added to both the resulting ifcOWL and BOT file. As with the resulting ifcOWL file, the WKT geometry added to the BOT file facilitates the alignment of the geometry with its corresponding building geometry defined in a different domain, interlinking the domains. Furthermore, the natural simplicity of BOT, when compared to ifcOWL, acts as an easier entry point to Linked Data BIM. The resulting BOT building from this project provides a more simple representation than its corresponding ifcOWL building. This could lead to the BOT representation being more widely used than its more complex ifcOWL representation.

3.4.2 Algorithm Design

The BOT building created from this solution defines the site, building, building storeys, spaces and walls only, with the site, building and walls having an assigned WKT geometry. There are no further complexities added to the building representation. The algorithm to create a BOT building with added WKT geometries contains the following steps:

1. Create BOT building component definitions
2. Create BOT building component relationship hierarchy

The resulting BOT building created in this solution contains five types of BOT building components:

1. BOT Site
2. BOT Building
3. BOT Storey
4. BOT Space
5. BOT Element

The extracted ifcOWL site and building resources are used to create the BOT Site and BOT Building. Each building storey that exists in the ifcOWL building maps to a BOT Storey. A building storey contains a number of rooms, these are defined as spaces in ifcOWL and BOT. All spaces in the ifcOWL building are modelled as a BOT Space. Every wall in the ifcOWL building is defined as a BOT Element (there is no explicit wall definition in BOT). The resulting BOT file from this process has a site, a building, building storeys, spaces (rooms) and elements (walls). Each of these building components are defined as RDF triples in the BOT file, using the same Unique Resource Identifiers (URIs) as defined in the original ifcOWL file.

WKT geometries are added to the BOT Site, BOT Building, and BOT Elements. The BOT Site is assigned the latitude and longitude geolocation that is extracted from the solution in Section 3.2. The BOT Building receives the same 2D WKT building footprint geometry as defined in Section 3.2. All BOT Elements (walls) are represented by a 2D WKT `LINESTRING`, defined by the start and end latitude and longitude coordinates of the wall. These WKT geometries provide a 2D GIS component to the BOT building, where a user can examine the building geometries through GeoSPARQL querying.

BOT facilitates the definition of relationships that define if one object contains another object using the *has* predicate definitions. This allows for the construction of a relationship hierarchy between BOT building components. The `hasBuilding` predicate relates a BOT Site to a BOT Building. The BOT `hasStorey` predicate relates a BOT Building to a BOT Storey. The BOT `hasSpace` predicate relates a BOT Storey to a number of BOT Spaces. The BOT `hasElement` predicate relates a BOT Space to a number of BOT Elements. A relationship hierarchy between BOT building components is constructed using these predicates, facilitating building component traversal. Figure 3.8 shows the hierarchy, where each element is related to a space, a space is related to a building storey, the building storey is related to the building and the building is related to the site, completing the hierarchy. The hierarchy created in the resulting BOT file provides a means of discovering how building components are associated to one another.

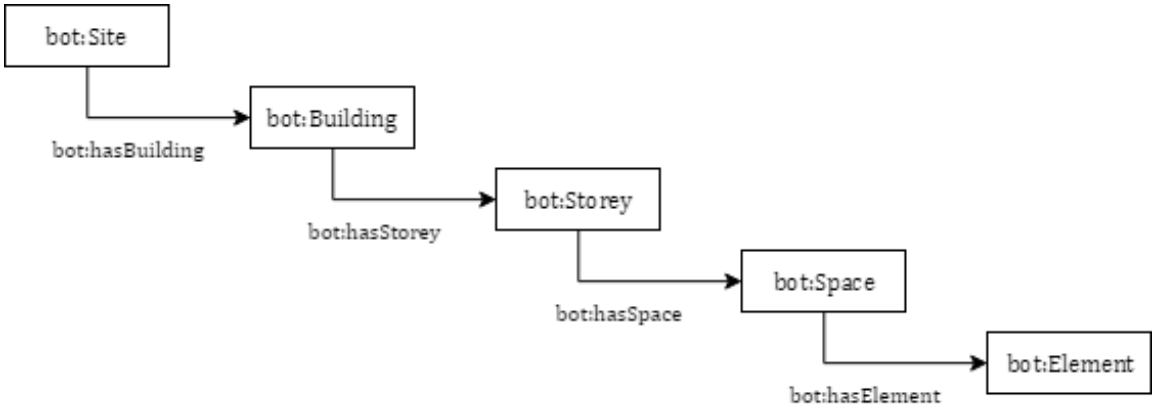


Figure 3.8: *Hierarchy of building components in the resulting BOT file. Note: A bot:Building would typically have multiple bot:Storeys, a bot:Storey would typically have multiple bot:Spaces and a bot:Space would typically have multiple bot:Elements.*

3.4.3 Algorithm Implementation

A BOT building defines the topology of the building structure. This solution extracts the primary ifcOWL building components for representation as BOT. The algorithm to carry out this process has two steps:

1. Create BOT building component definitions
2. Create BOT building component relationship hierarchy

The implementation of these two steps is described in Sections 3.4.3.1 and 3.4.3.2 below. This solution relies heavily of the Apache Jena framework. Jena facilitates the extraction of RDF triples from the original ifcOWL file and definition of new RDF triples in the BOT file.

3.4.3.1 Create BOT Building Component Definitions

The solution implemented to define a BOT building representation first creates three physical building components, the BOT Site, the BOT Building and BOT Elements. The RDF resources that represent an `IfcSite`, an `IfcBuilding` and multiple `IfcWallStandardCase` objects are extracted from the ifcOWL building to create the three BOT components. The URIs for these resources are used as the URIs for the corresponding BOT definitions. The algorithm described in Section 3.2 defines a geolocation WKT `POINT` for the site geometry and a WKT `MULTILINESTRING` for the building footprint geometry. These are used to represent the BOT Site and BOT Building geometries as GeoSPARQL `hasGeometry` properties. Section 3.2 also describe how each building wall has a WKT `LINESTRING` geometry, defined by two latitude and

longitude endpoint coordinates. This geometry is added to each BOT Element definition as a GeoSPARQL `hasGeometry` property. Figure 3.9 below gives example BOT Site and BOT Building definitions with their associated geometries.

```
<http://linkedbuildingdata.net/ifc/resources20170627_105709/IfcSite_389>
        a                bot:Site ;
        geo:hasGeometry  <urn:geom:pt:0KMpiAlnb52RgQuM1CwVfd>.

<urn:geom:pt:0KMpiAlnb52RgQuM1CwVfd>
        geo:asWKT  " POINT (8.436539 49.100435)"^^geo:wktLiteral.

<http://linkedbuildingdata.net/ifc/resources20170627_105709/IfcBuilding_434>
        a                bot:Building ;
        geo:hasGeometry  <urn:geom:pt:2hQBAVPOr5VxhS3Jl0O47h> .

<urn:geom:pt:2hQBAVPOr5VxhS3Jl0O47h>
        geo:asWKT  "MULTILINESTRING ((8.436620415013731 49.10045985664715,
8.436665972347003 49.10039589031979) , (8.436620415013731 49.10045985664715,
8.436665972347003 49.10039589031979) , (8.436665972347003 49.10039589031979,
8.43658455733327 49.10037103364061) , (8.436665972347003 49.10039589031979,
8.43658455733327 49.10037103364061) , (8.436539 49.10043500000003,
8.436620415013731 49.10045985664715) , (8.436539 49.10043500000003,
8.436620415013731 49.10045985664715) , (8.43658455733327 49.10037103364061,
8.436539 49.10043500000003) , (8.43658455733327 49.10037103364061, 8.436539
49.10043500000003) )"^^geo:wktLiteral .
```

Figure 3.9: *Example BOT Site and BOT Building definitions. Each component has a WKT geometry defined using the GeoSPARQL hasGeometry predicate.*

Resulting from this process of defining the BOT Site, BOT Building and BOT Elements is a BOT representation off three physical building components with their associated geometries. However, these components live in isolation within the BOT model. A notion of how the building components relate to one another is needed to complete the BOT representation of an ifcOWL building.

3.4.3.2 Create BOT Building Component Relationship Hierarchy

The BOT building component relationship hierarchy defines how components relate to one another. This facilitates traversal of the hierarchy through SPARQL querying, allowing the user to explore related BOT building components. The relationship definitions rely heavily on how they are defined in the ifcOWL building. The ifcOWL version used to define the ifcOWL building impacts how ifcOWL components relate to one another. As discussed above in Section 3.3.3.2, the test data for this project saw the use of both the ifcOWL `IFC4_ADD1` ontology and the ifcOWL `IFC2X3_TC1` ontology. Each ifcOWL version uses a different RDF predicate to define a relationship between two objects. This is discussed in more detail below.

The BOT hierarchy, as shown in Figure 3.8 above, has the site at the top of the hierarchy. A BOT building is related to a BOT Site by using the BOT `hasSite` predicate. This relationship is constructed using the BOT Site and BOT Building resources as defined in Section 3.4.3.1 above.

Next in the hierarchy is the relationship between a BOT Building and a BOT Storey. The Java Jena program extracts all `IfcBuildingStorey` resources from the ifcOWL building. The URIs of these resources are used to create BOT Storeys. The relationship between the BOT Building and each BOT Storey is defined using the BOT `hasStorey` predicate. This relationship is created for each building storey in the ifcOWL building.

Each storey has multiple spaces, or rooms, associated with it. The ifcOWL `IFC4_ADD1` property `relatingObject_IfcRelAggregates` and the `IFC2X3_TC1` property `relatingObject_IfcRelDecomposes` define that an object is related to another object. In this case, an ifcOWL building storey is related to a number spaces (rooms) contained within the building storey. The spaces are related to the storey through definition of the `IFC4_ADD1` `relatedObjects_IfcRelAggregates` predicate and definition of the `IFC2X3_TC1` `relatedObjects_IfcRelDecomposes` predicate. All spaces relating to a building storey can be extracted using these predicates. Figure 3.10 below shows how an ifcOWL building storey is related to a number of ifcOWL spaces through use of the ifcOWL `IFC4_ADD1` predicates `relatingObject_IfcRelAggregates` and `relatedObjects_IfcRelAggregates`. Similar predicate definitions are also used in the ifcOWL `IFC2X3_TC1` version.

In BOT, a BOT space is related to a BOT Storey using the `hasSpace` predicate. A BOT Space is created for each ifcOWL space. These spaces are then related to their parent BOT Storey using the BOT `hasSpace` predicate. The BOT relationship hierarchy now has a site that relates to a building, a building that relates to multiple storeys and storeys that relate to multiple spaces. The final step in creating the hierarchy is to relate the BOT Elements to their associated spaces.

```
inst:IfcRelAggregates_20914
        ifcowl:globalId_IfcRoot      inst:IfcGloballyUniqueId_71746 ;
        ifcowl:ownerHistory_IfcRoot  inst:IfcOwnerHistory_12 ;
        ifcowl:relatingObject_IfcRelAggregates  inst:IfcBuildingStorey_479 ;
        ifcowl:relatedObjects_IfcRelAggregates  inst:IfcSpace_20909 ;
        ifcowl:relatedObjects_IfcRelAggregates  inst:IfcSpace_21283 ;
        ifcowl:relatedObjects_IfcRelAggregates  inst:IfcSpace_21640 ;
        ifcowl:relatedObjects_IfcRelAggregates  inst:IfcSpace_33774 ;
        ifcowl:relatedObjects_IfcRelAggregates  inst:IfcSpace_34191 ;
        ifcowl:relatedObjects_IfcRelAggregates  inst:IfcSpace_34763 .
```

***Figure 3.10**: Example snippet from an ifcOWL file showing how an IfcBuildingStorey relates to multiple IfcSpaces.*

Creating a relationship between a BOT Space and a number of BOT Elements requires examination of how an ifcOWL wall is related to an ifcOWL space. The ifcOWL `relatingSpace_IfcRelSpaceBoundary` predicate defines how an ifcOWL space is associated with an ifcOWL `IfcRelSpaceBoundary` resource definition. The `IfcRelSpaceBoundary` resource defines a how objects relate to the boundary of the space through use of the `relatedBuildingElement_IfcRelSpaceBoundary` predicate. In this case, the predicate relates a wall to the boundary of a space. Figure 3.11 below gives a snippet from an ifcOWL file that relates a wall to a space. A BOT `hasElement` relationship can now be created based off this ifcOWL space to ifcOWL wall relationship. The BOT `hasElement` predicate is used to relate a BOT Space to all its associated boundary BOT Elements (walls). This is done for all BOT Spaces that exist in the BOT building representation.

```
inst:IfcWallStandardCase_15042
    rdf:type  ifcowl:IfcWallStandardCase .

inst:IfcSpace_20909
    rdf:type  ifcowl:IfcSpace .

inst:IfcRelSpaceBoundary_76511
    ifcowl:relatingSpace_IfcRelSpaceBoundary  inst:IfcSpace_20909 .
    ifcowl:relatedBuildingElement_IfcRelSpaceBoundary  inst:IfcWallStandardCase_15042 .
```

Figure 3.11: *Example snippet from an ifcOWL file showing how an IfcWallStandardCase is related to an IfcSpace. This relationship definition is present for all walls relating to a space.*

The hierarchy is now complete, facilitating the traversal of the BOT building through all its structural building components. Figure 3.12 below gives an example snippet of the resulting BOT file, showing how all BOT building components relate to one another. The WKT geometries have been excluded from this example. All resources in this example have a namespace in the form "http://linkedbuildingdata.net/ifc/resources20170918_190646", which have been reduced to "ifc_res/" to conserve space. Only one BOT Space and one BOT Element is given is this example.

```
<ifc_res/IfcSite_389>
       a                bot:Site ;
       geo:hasGeometry  <urn:geom:pt:0KMpiAlnb52RgQuM1CwVfd> ;
       bot:hasBuilding  < ifc_res/IfcBuilding_434> .

<ifc_res/IfcBuilding_434>
       a                bot:Building ;
       geo:hasGeometry  <urn:geom:pt:2hQBAVPOr5VxhS3Jl0O47h> ;
       bot:hasStorey    <ifc_res/IfcBuildingStorey_479> ;
       bot:hasStorey    <ifc_res/IfcBuildingStorey_35065> .

<ifc_res/IfcBuildingStorey_479>
       a             bot:Storey ;
       bot:hasSpace  <ifc_res/IfcSpace_21283> ;
       bot:hasSpace  <ifc_res/IfcSpace_34191> ;
       bot:hasSpace  <ifc_res/IfcSpace_34763> ;
       bot:hasSpace  <ifc_res/IfcSpace_20909> ;
       bot:hasSpace  <ifc_res/IfcSpace_33774> ;
       bot:hasSpace  <ifc_res/IfcSpace_21640> .

<ifc_res/IfcSpace_21283>
       a               bot:Space ;
       bot:hasElement  <ifc_res/IfcWallStandardCase_18698> ;
       bot:hasElement  <ifc_res/IfcWallStandardCase_18465> ;
       bot:hasElement  <ifc_res/IfcWallStandardCase_17040> ;
       bot:hasElement  <ifc_res/IfcWallStandardCase_32407> .

<ifc_res/IfcWallStandardCase_18698>
       a                bot:Element ;
       geo:hasGeometry  <urn:geom:pt:3rPX_Juz59peXXY6wDJl18> .
```

***Figure 3.12**: Resulting BOT building representation. All BOT building components are related through the BOT relationship hierarchy.*

3.5 GeoSPARQL Building Footprint Alignment

3.5.1 Motivation

The integration of BIM and GIS is the ultimate goal of this project. Sections 3.2 and 3.4 discuss solutions that integrate 3D BIM and 2D GIS through the extraction and definition of 2D WKT geometries. These geometries can be used in 2D GIS applications to explore the buildings in a 2D GIS context. A process is needed to align geometries with the same building geometries defined in a different domain. This would enable the interlinking of a building with other domains, enriching the knowledge available on the building. The fourth and final solution implemented in this project defines a process to align 2D WKT building footprint geometries through GeoSPARQL querying.

The project solutions to integrate BIM and 2D GIS, defined in Sections 3.2 and 3.4, have resulted in an updated ifcOWL file and a new BOT file with added 2D WKT geometries. These files are loaded into a RDF triple store whereby the BIM data is made available for querying. A querying procedure currently does not exist to extract and align BIM data WKT geometries. It was therefore necessary to create such a process, completing the solutions toolkit.

3.5.2 Algorithm Design

GeoSPARQL, a spatial extension of the SPARQL query language, provides a number of topological relationship functions. These functions enable the discovery of relationships between 2D GIS geometries, providing capability for geometry alignment. Building alignment using a GeoSPARQL function in isolation does not give a reliable and consistent solution. A second matching property is therefore desirable to improve the accuracy of geometry alignment. The GeoSPARQL building alignment solution implemented in this project uses a GeoSPARQL matching function and the area of the WKT polygon geometry as the second matching property. The input building file must first be processed to determine the polygon area. The entire matching procedure contains the following steps:

1. Determine total area of WKT geometry
2. Attach area value to building representation
3. GeoSPARQL querying of the processed building data

The test data used in this project did not contain the polygon area value for each WKT building footprint geometry. Pre-processing of the test data is necessary to add the polygon area value to buildings before querying. A program using the Jena Java framework processes the building data to calculate the polygon area. The surveyor's area formula [32] is implemented in Java to calculate polygon area. This formula requires the polygon points to be stored in a sequential counter-clockwise formation. The resulting area value is attached back to the building representation as an RDF triple. The processed building representation is now ready for GeoSPARQL building footprint alignment.

A number of GeoSPARQL topological relations exist that allow for the discovery of relationships between geometries, and therefore resources, in a RDF graph. The following geometric relations can be expressed in a GeoSPARQL query: equality, disjointness, intersection, touches, within, contains and overlaps. This project focused on the use of the GeoSPARQL function `geof:sfOverlaps`, a function that matches geometries that partially overlap [8]. The processed building data, with added geometry area value, is stored in a triple store and queried via the SPARQL endpoint. The YASGUI [33] query interface draws WKT geometries in a GIS context. The resulting aligned geometries, from GeoSPARQL querying of the triple store, can be seen via the YASGUI *Geo* results page, verifying the solution. Figure 3.13 shows a screenshot of the YASGUI *Geo* results page. The green and red geometries represent two aligned buildings that are shown as a result of the building matching filter criteria described above.

Figure 3.13: *Screenshot of the YASGUI Geo results page showing two aligned building geometries, drawn in green and red, next to a road.*

3.5.3 Algorithm Implementation

As discussed in Section 3.5.2 above, the algorithm to align a GeoSPARQL WKT building geometry uses a GeoSPARQL function and the area of the polygon geometry as a matching property. The algorithm follows three steps, which are discussed individually in Sections 3.5.3.1, 3.5.3.2 and 3.5.3.3:

1. Determine total area of WKT geometry
2. Attach area value to building representation
3. GeoSPARQL querying of the processed building data

The test data for this solution uses an ifcOWL and BOT representation of the same building to simulate alignment. The WKT geometry of the BOT building is manually altered so that it is slightly different from the ifcOWL building's WKT geometry. This simulated altering of WKT geometries was necessary as I did not have access to corresponding buildings of ifcOWL test data. Ideally I would have had corresponding representations of the ifcOWL buildings defined in a different dataset, facilitating the geometry alignment of the ifcOWL files produced from this project with other definitions of the same building.

3.5.3.1 Determine Total Area of WKT Geometry

Pre-processing of ifcOWL data is necessary to add the geometry area value to the building before querying. A program using Apache Java [12] processes the building data to calculate each WKT polygon area. The program first searches the input ifcOWL building file for the `IfcBuilding`

definition and searches the input BOT building for the BOT `Building` definition. These both have a GeoSPARQL `hasGeometry` property, storing a URI that represents a geometry. A WKT `POLYGON` is extracted from the GeoSPARQL `asWKT` property of the URI. The extracted WKT polygon contains a number of latitude and longitude coordinates that define the external wall building footprint. All points are extracted from the WKT geometry and transformed to Cartesian xy-coordinates using the WGS 84 Web Mercator (EPSG:3857) projection [47]. The xy-coordinates are stored in a counter-clockwise formation, and are used to calculate the polygon area of a building. The surveyor's area formula [32] is implemented in the Java program to calculate polygon area. The resulting area value for each building is defined in metres squared and is ready to be added back to the respective ifcOWL `IfcBuilding` and BOT `Building` definitions.

3.5.3.2 Attach Area Value to Building Representation

Determination of the WKT geometry area is a necessary step to achieve accurate building alignment. The area value acts as a second matching property, improving the accuracy of the solution. The calculated geometry area value is added back to the original `IfcBuilding` and BOT `Building` resources as a numeric double literal. A `hasArea` predicate relates the resources to the area literal value. The ifcOWL file is now ready for querying. The processed data is stored on a Parliament triple store [49]. Parliament stores RDF triple definitions as a connected graph, providing an endpoint for querying of the data. The YASGUI [33] query interface is used in this project to query data via the Parliament endpoint.

3.5.3.3 GeoSPARQL Querying of the Processed Building Data

Querying of the test data makes use of the SPARQL `Filter` function, which returns results based on filter conditions. This solution specifies two filter conditions for building alignment: that building WKT geometries overlap (using the GeoSPARQL `geof:sfOverlaps` function) and that the calculated area of one WKT polygon is within +/- 25% the area of the overlapping polygon. Both aligned polygons, representing the same physical building, are returned if these two filter conditions are met. Figure 3.14 shows an example SPARQL query to align two buildings. An ifcOWL building is aligned with its BOT building equivalent. The filter conditions at the end of the query specify the GeoSPARQL overlapping condition and the geometry area matching condition.

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC4_ADD1#>
PREFIX bot: <https://w3id.org/bot#>

SELECT ?building1 ?building2 ?buildingWkt1 ?buildingWkt2 ?area1 ?area2
WHERE {
       ?building1 a ifcowl:IfcBuilding.
       ?building1 geo:hasGeometry ?buildingGeom1.
       ?buildingGeom1 geo:asWKT ?buildingWkt1.
       ?building1 geo:hasArea ?area1.

       ?building2 a bot:Building.
       ?building2 geo:hasGeometry ?buildingGeom2.
       ?buildingGeom2 geo:asWKT ?buildingWkt2.
       ?building2 geo:hasArea ?area2.

       FILTER(geof:sfOverlaps(?buildingWkt1, ?buildingWkt2)).
       FILTER((xsd:double(?area2)     >     (xsd:double(?area1)*     0.75))     &&
(xsd:double(?area2) < (xsd:double(?area1)* 1.25))).
} limit 1
```

Figure 3.14: *The SPARQL used to align two overlapping building geometries. The two buildings and their geometries are returned only if the two filter conditions are met.*

4. Evaluation

This chapter focuses on providing an evaluation of the four solutions implemented as part of the solutions toolkit. Three sections are provided, Sections 4.1, 4.2 and 4.3. Section 4.1 details the results obtained for each solution, presenting images of building geometries and code snippets. Section 4.2 evaluates the algorithm design of each solution and discusses the limitations of the solution. Section 4.3 concludes this chapter, discussing some of the challenges that were faced during this project. An evaluation of the implemented solutions sets up an opportunity to reflect on how well each solution works towards the project goal of integrating BIM and GIS buildings.

4.1 Results

This section presents the results obtained for each of the four solutions in the solutions toolkit. A subsection is provided for each solution in subsections 4.1.1, 4.1.2, 4.1.3 and 4.1.4, showing images and screenshots of the results obtained for that solution.

A Note on Test Data

The BIM test data used in this project was obtained from an online IFC and ifcOWL repository located at [43]. Six ifcOWL buildings were downloaded from this website to aid in designing, implementing and testing the implemented solutions. The chosen files are given in Table 4.1 below, along with their RDF triple count, wall count and number of building storeys. These six files were chosen due to their compatibility with the implemented solutions. This project only extracts walls of the type `IfcWallStandardCase` from ifcOWL buildings. It was therefore necessary that the test data contained these walls. At the time of implementing this project there was only a small number of models available online that contained `IfcWallStandardCase` walls. Furthermore, there was a shortage of ifcOWL models available that were models of complete and comprehensive buildings. The online repository contained models that were incomplete, models that were not building structures and models that did not contain walls of the type `IfcWallStandardCase`. The six ifcOWL files chosen for this project met the required criteria.

The chosen ifcOWL files have varying complexity, in terms of RDF triple count, and in terms of the number of walls represented in the model. This range of complexity ensured that the

implemented solutions were somewhat robust, as robust as they could be with the shortage of test data. The RDF triple count, wall count and number of building storeys of each file is given in Table 4.1, indicating the level of complexity of the ifcOWL files. The wall count is the number of `IfcWallStandardCase` walls and the building storey count is the number of building storeys that contain walls of the type `IfcWallStandardCase`.

File Name	RDF Triple Count	Wall Count	Building Storey Count
`smallhouse_saref.tll`	100246	6	1
`Duplex_A_20110907_optimized.ttl`	205141	56	4
`Simple3-storeytestbuilding.ttl`	222545	21	3
`20170804_Musterhaus_MIT.ttl`	239899	24	3
`AC20-FZK Haus.ttl`	275685	13	2
`Barcelona_Pavilion.ttl`	1033905	35	2

Table 4.1: *The six ifcOWL files downloaded from the online ifcOWL repository for use in this project.*

From Table 4.1 it can be seen that there are varying levels of complexity. This complexity can be measured as the number of walls and building storeys that need to be extracted from the file or the level of detail of representation (i.e. number of RDF triples). It can be said that the files `Simple3-storeytestbuilding.ttl` and `20170804_Musterhaus_MIT.ttl` have a similar level of complexity. They contain a similar number of RDF triples, walls and building storeys. The file `AC20-FZK Haus.ttl` is the most comprehensive file out of all the buildings. It only has 13 walls, yet it has the second largest RDF triple count, indicating the level of detail used to describe the building's components. This high level of detail allows for more intricate testing of the solutions. As a result, this file allowed for the correct extraction of an external wall 2D WKT geometry, a complete fully connected 3D OBJ model, a BOT representation with fully connected relationship hierarchy and it was the file used to test the GeoSPARQL building alignment solution. Ideally, a larger set of complete, comprehensive and suitable ifcOWL buildings would have been used to test the solutions in this project. However, the sensitivity issues around BIM data and the complexities associated with developing IFC and ifcOWL result in a small number of suitable files being freely available online.

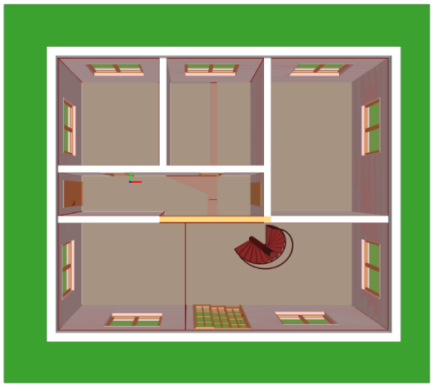
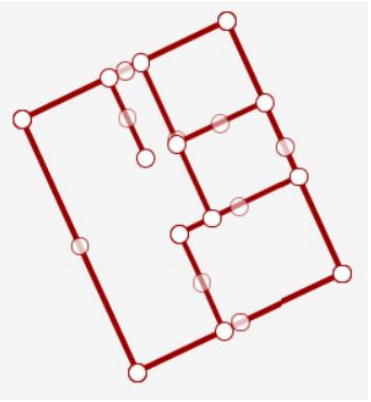
A Note on Result Presentation

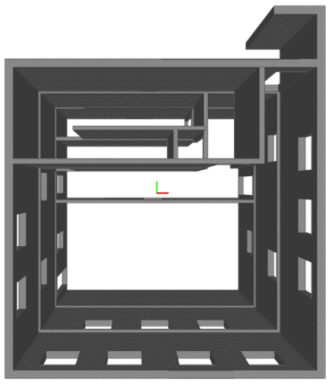
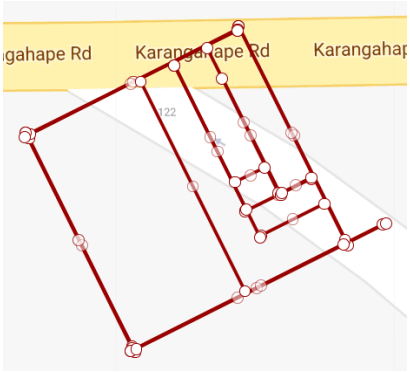
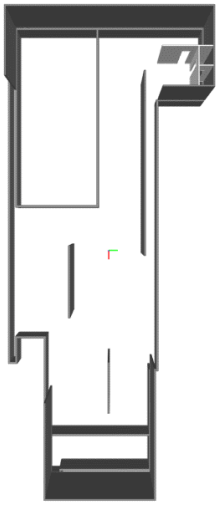
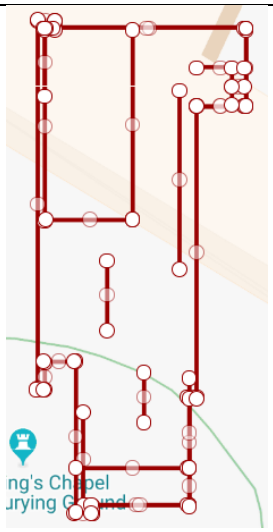
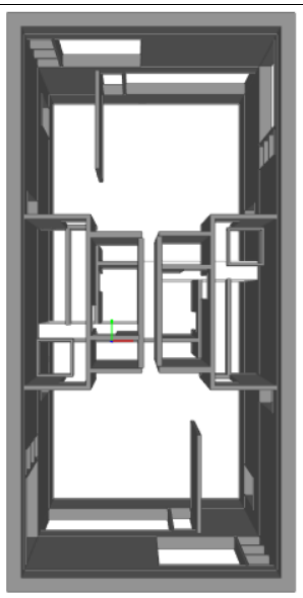
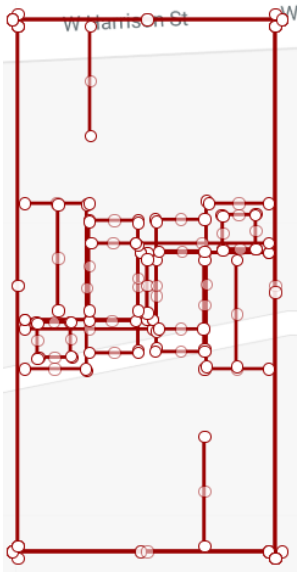
The results presented for the first two implemented solutions contain images of the resulting geometries and their ground truth IFC models. These are screenshots of the geometries as viewed

in their respective viewing software. Screenshots of the IFC models are the top-down view of the model as viewed in BIM Vision [45], a free IFC model viewer. The images for the 2D WKT GIS representations are screenshots of the resulting 2D WKT geometries, taken from an online WKT Wicket [46]. The images depicting 3D OBJ building models are screenshots of the resulting models captured in Blender [44]. Blender is a free 3D modelling software package that allows the importation and viewing of 3D OBJ models.

4.1.1 ifcOWL Building Footprint Representation as 2D WKT Results

Table 4.2 below shows the results for the solution that extracts a 2D WKT building footprint geometry from a 3D BIM building. The file name of each ifcOWL file is given in Table 4.2, along with its top-down BIM model view and the resulting 2D WKT geometry. As mentioned in Section 3.2, this solution extracts the `IfcWallStandardCase` instances of walls only, representing them as a 2D WKT geometry. The 3D BIM images shown in Table 4.2 show the building's wall structure of instances of the wall type `IfcWallStandardCase`. Each drawn 2D WKT building geometry contains both internal and external walls of all building storeys, represented as a WKT `MULTILINESTRING`. Whereas in the GeoSPARQL building alignment solution the external walls are represented only, drawn as a WKT `POLYGON`. Finally, note that some of the buildings are rotated at an angle are some are not. The buildings rotated at an angle contained a true north angle whereas the other buildings did not contain such an angle and are defaulted to a +90 degrees rotation.

ifcOWL File Name	3D IFC BIM Representation	2D WKT GIS Representation
`AC20-FZK-Haus.ttl`		

Simple3-storeytestbuilding.ttl		
Barcelona_Pavilion.ttl		
Duplex_A_20110907_optimized.ttl		

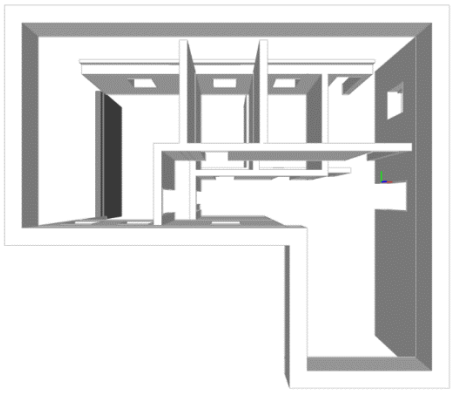
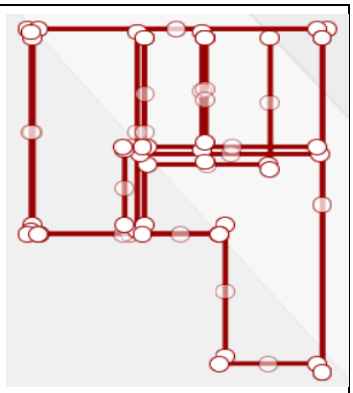
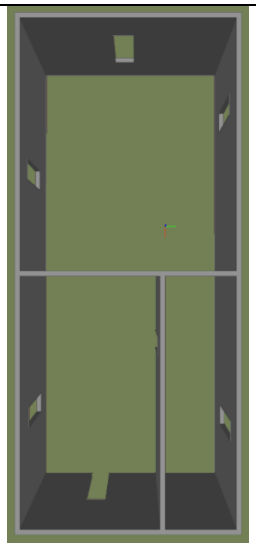
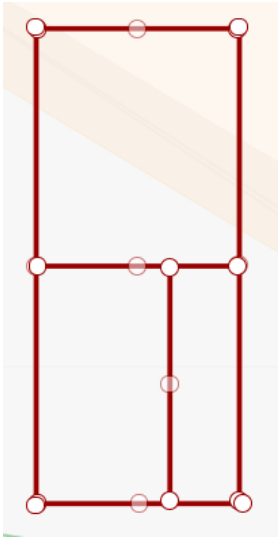
`20170804_Musterhaus_MIT.ttl`		
`smallhouse_saref.ttl`		

Table 4.2: *Results of the solution to represent 3D BIM buildings as 2D WKT GIS geometries. Both internal and external walls of each building are shown.*

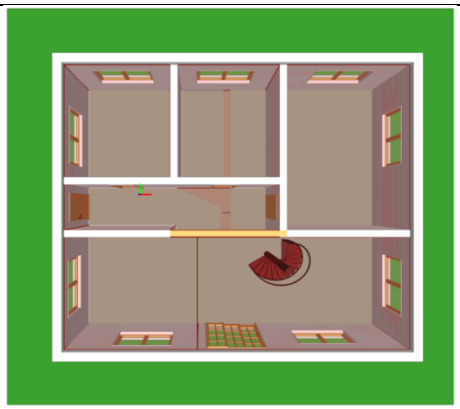
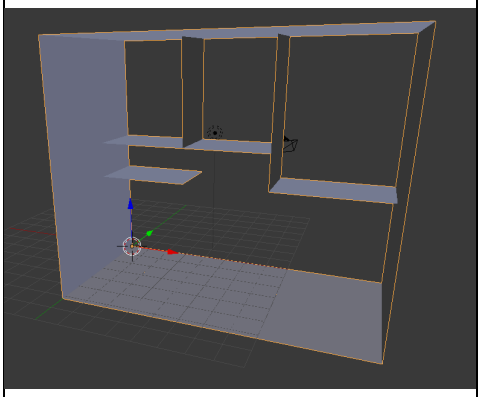
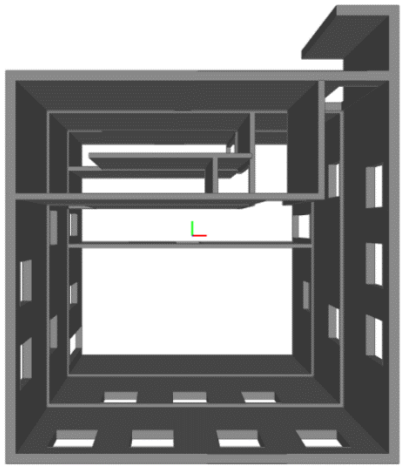
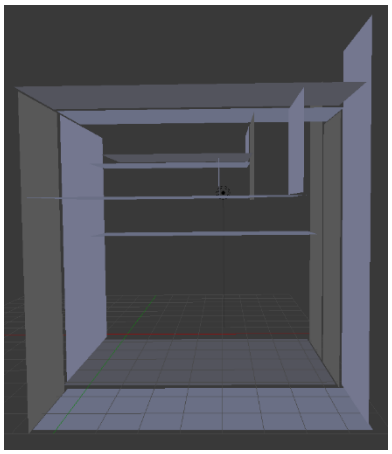
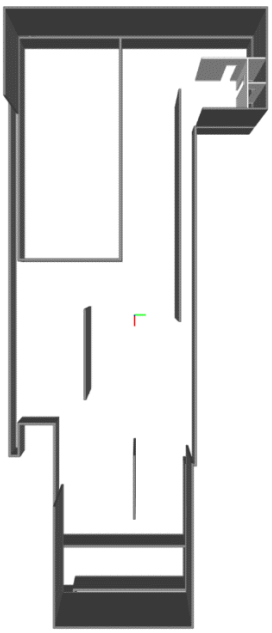
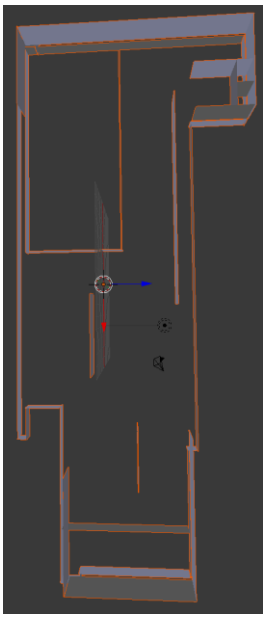
The 2D WKT geometries in Table 4.2 above are attached to their original source ifcOWL files. The geolocation of these buildings are also attached as a WKT `POINT`. These ifcOWL files, with added WKT geometries, are stored in a Parliament RDF triple store for querying. GeoSPARQL querying of the data extracts the WKT geometries for representation on a 2D map. The YASGUI query interface was used in this project to query and view the processed ifcOWL data. Figure 4.1 below shows the query results, giving the WKT building footprint geometry and WKT point geolocation of the ifcOWL file `Barcelona_Pavilion.ttl`, as drawn by the YASGUI *Geo* results page. The SPARQL and GeoSPARQL query code is also given.

```
PREFIX ifcowl: <http://www.buildingsmart-tech.org/ifcOWL/IFC2X3_TC1#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
SELECT ?building ?buildingWkt ?site ?siteWkt
WHERE {
        ?building a ifcowl:IfcBuilding .
        ?building1 geo:hasGeometry ?buildingGeom .
        ?buildingGeom1 geo:asWKT ?buildingWkt .
        ?site a ifcowl:IfcSite .
        ?site geo:hasGeometry ?siteGeom .
        ?siteGeom geo:asWKT ?siteWkt .
}
```

Figure 4.1: *Image on the right is the drawn WKT building footprint geometry and WKT geolocation point of Barcelona_Pavilion.ttl, as seen in the YASGUI Geo results page. These WKT geometries are drawn as a result of the given SPARQL code.*

4.1.2 ifcOWL Building Footprint Representation as 3D OBJ Results

Table 4.3 below shows the results for the solution that extracts a 3D OBJ building footprint model from a 3D BIM building. The file name of each ifcOWL file is given in Table 4.3, along with its top-down BIM model view and the resulting 3D OBJ building model. The resulting OBJ building model is a representation of the building walls as they are defined in the original ifcOWL model, with each OBJ building model containing both internal and external walls from all building storeys. The images show the resulting 3D OBJ models as viewed in Blender, with some images taken at an angle to give depth to the image.

ifcOWL File Name	3D IFC BIM Representation	3D OBJ Representation
AC20-FZK-Haus.ttl		
Simple3-storeytestbuilding.ttl		
Barcelona_Pavilion.ttl		

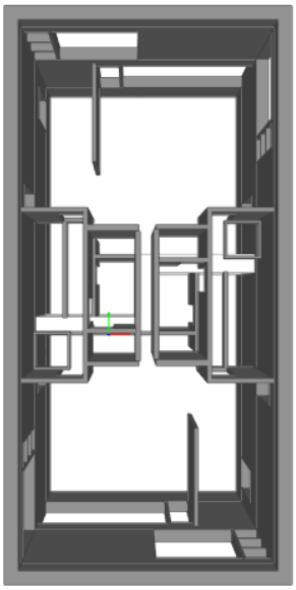
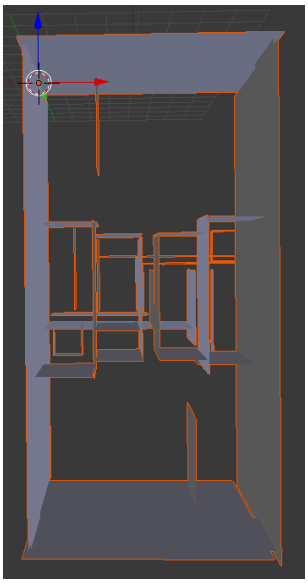
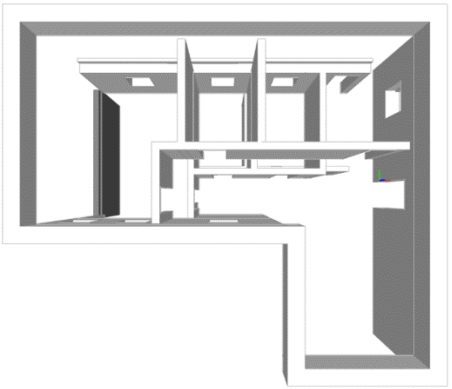
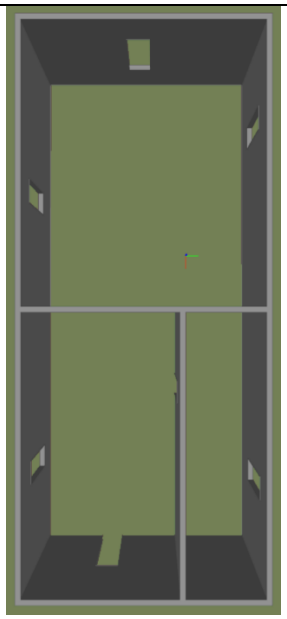
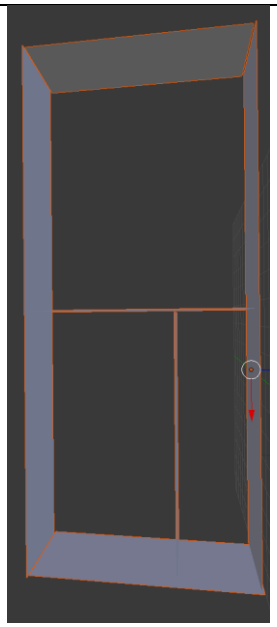
Duplex_A_20110907_optimized.ttl		
20170804_Musterhaus_MIT.ttl		
smallhouse_saref.ttl		

Table 4.3: *Results of the solution to represent 3D BIM buildings as 3D OBJ models. Both internal and external walls from all building storeys are shown.*

4.1.3 ifcOWL to Building Topology Ontology (BOT) Results

The solution to create a BOT building representation of an ifcOWL building generates a BOT file with added structural building component WKT geometries. This BOT file follows a relationship hierarchy, the implementation of which is discussed in Section 3.4.3.2 above. The BOT Site sits at the top of the hierarchy, followed by the BOT Building, BOT Storeys, BOT Spaces and, at the bottom of the hierarchy, the BOT Elements. This hierarchy can be seen when inspecting the resulting BOT files from this solution.

The BOT files for all the ifcOWL test data will not be shown in this document to conserve space. However, Appendix B shows a snippet of BOT RDF triples for the input ifcOWL file `Simple3-storeytestbuilding.ttl`. Given in Appendix B is the BOT Site, the BOT Building, a BOT Storey, a BOT Space and a BOT Element. Also given is the WKT geometries for the BOT Site, BOT Building and the BOT Element. Inspection of Appendix B shows the BOT building component relationship hierarchy using the BOT *has* predicates. Note, all resources in the example in Appendix B have a namespace in the form "http://linkedbuildingdata.net/ifc/resources20170627_105707", which has been reduced to "ifc_res/" to conserve space.

Figure 4.2 below shows the SPARQL query code used to access each BOT building component in the BOT file resulting from `Simple3-storeytestbuilding.ttl`. All wall (BOT Element) WKT `LINESTRING` geometries are also shown in Figure 4.2. These are extracted using GeoSPARQL and are drawn in the *Geo* results page of the YASGUI query interface. The combination of all wall geometries create the overall 2D GIS building footprint. Excluded from the example in Figure 4.2 is the GeoSPARQL extraction of the BOT Site WKT `POINT` geolocation and the BOT Building WKT `MULTILINESTRING` geometry. Note, the geometry of the building has been scaled up as there is not a high zoom level on the YASGUI *Geo* results page.

```
PREFIX bot: <https://w3id.org/bot#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

SELECT ?site ?building ?storey ?space ?element ?wallWktLinestring
WHERE {
       ?site a bot:Site.
       ?site bot:hasBuilding ?building.
       ?building bot:hasStorey ?storey.
       ?storey bot:hasSpace ?space.
       ?space bot:hasElement ?element.
       ?element geo:hasGeometry ?wallGeom.
       ?wallGeom geo:asWKT ?wallWktLinestring.
}
```

Figure 4.2: *The SPARQL code to extract BOT building components using the BOT building component relationship hierarchy. Also given is the WKT wall (BOT Element) LINESTRING geometries, combining to give the building footprint geometry. This image of the WKT geometries is a screenshot taken from the Geo results page of the YASGUI query interface after executing the given SPARQL code.*

4.1.4 GeoSPARQL Building Footprint Alignment Results

The YASGUI query interface is used to execute the GeoSPARQL building alignment solution and examine its results. An ifcOWL building and its corresponding BOT building are loaded into a Parliament RDF triple store. Both the ifcOWL file and BOT file represent the same building, however, the BOT Building WKT geometry has been altered by rotating the geometry a few degrees. This is to simulate the building alignment of an ifcOWL building with its corresponding building defined in a different dataset. The query attempts to align the buildings based on the GeoSPARQL `geof:sfOverlaps` function and if the building geometry area values are within +/-25% of each other. If both these conditions are met the overlapping buildings are drawn in the *Geo* results page of the YASGUI interface.

Figure 4.3 below gives a screenshot of the query used to align an ifcOWL building with its corresponding BOT building, based on the two matching conditions. This query is performed on the `AC20-FZK-Haus.ttl` ifcOWL file, after adding WKT geometries, and the BOT version of this file, after BOT processing with added WKT geometries. Also given in Figure 4.3 is the resulting aligned geometries, as seen in the YASGUI *Geo* results page. Colouring of the polygons identify the ifcOWL building from the BOT building. The red geometry represents the ifcOWL building and the green geometry represents the BOT building. Note, the geometries shown are quite small

(YASGUI *Geo* does not permit a high zoom level), making it difficult to differentiate between the red ifcOWL geometry and the green BOT geometry. The green BOT geometry is a rotated version of the ifcOWL geometry and overlays the red ifcOWl geometry. A sliver of red can be seen on the right hand side of the overlapping geometries, representing the underlying ifcOWL building.

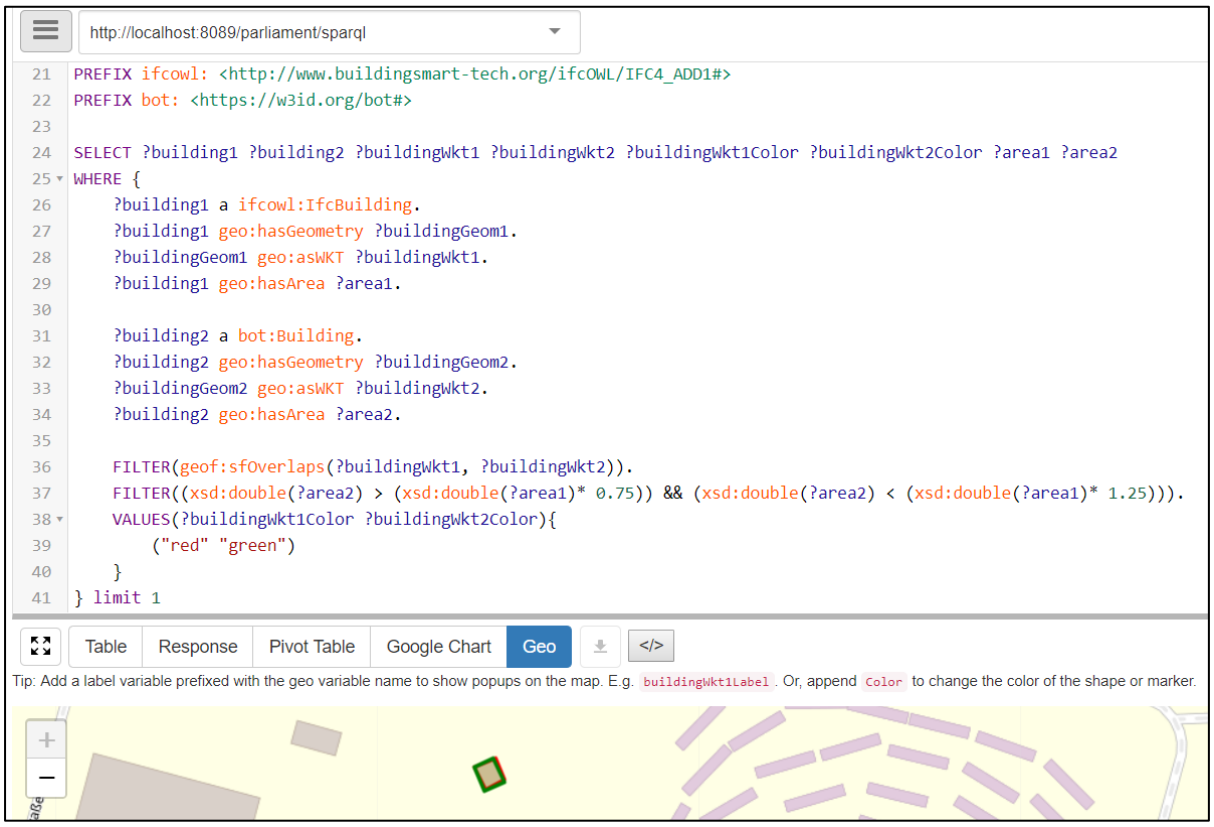


Figure 4.3: *The SPARQL and GeoSPARQL used to align two overlapping building geometries. Screenshot is taken from YASGUI, where the Geo results page draws the ifcOWL (red) and BOT (green) building geometries.*

4.2 Evaluation of the Solutions Toolkit

This section gives an evaluation on the algorithms used to create the four implemented solutions. An evaluation is provided on the usability of the solutions and the quality of their obtained results. A discussion is also provided on the limitations associated with each solution. Four subsections are provided, with each subsection evaluating one of the four implemented solutions:

1. ifcOWL building footprint representation as 2D WKT
2. ifcOWL building footprint representation as 3D OBJ
3. ifcOWL to Building Topology Ontology (BOT)
4. GeoSPARQL building footprint alignment

4.2.1 Evaluation of ifcOWL Building Footprint Representation as 2D WKT

The process to extract and represent the 2D WKT building footprint of an ifcOWL building forms the basis of all solutions implemented as part of this project. It provides an initial wall layout for the 3D OBJ building model, provides geometries for BOT building representations and provides a geometries that can be queried with GeoSPARQL for building alignment. The solutions toolkit therefore relies heavily on the process to represent a 3D BIM model as a 2D WKT geometry. This process can be evaluated on the accuracy and usability of the resulting geometries, the performance of the solution and the limitations of the solution.

4.2.1.1 Accuracy and Usability of Results

The primary evaluation that can be performed on this solution is to examine the resulting WKT geometries in Table 4.2 above. The WKT geometries can be evaluated visually, comparing them to their IFC models as seen in BIM Vision. This is clearly not the most desirable evaluation method, since no accuracy metric is produced, although visually comparing the two building representations gives an instantaneous and clear indication of how well a WKT geometry represents its BIM IFC model.

It can be seen from the table that the drawn WKT geometries are an accurate representation of the wall structure of their parent IFC models. Walls are positioned in their correct locations and have dimensions that match the parent IFC model. Intricate modes, such as the `Barcelona_Pavilion.ttl` and `Duplex_A_20110907_optimized.ttl` have a corresponding intricate 2D WKT geometry.

However, the drawn WKT geometries contain both internal and external walls from all building storeys. A 2D GIS geometry represents a building as a 2D building footprint outline, disregarding internal walls and typically representing the building's highest level or roof of the building. Figure 4.4 below gives a screenshot that was taken from Google Maps, depicting 2D GIS building geometries. The 2D building geometries in Google Maps give an overall external wall building footprint outline, representing the space a building fills regardless of building storey.

Figure 4.4: *Screenshot taken from Google Maps showing 2D building footprint geometries.*

The resulting extracted 2D WKT geometries in this solution should therefore represent the external walls of the IFC building model only, providing similar geometries to those in Google Maps. They would then match the 2D GIS standard of building geometry representation. During the course of this project an attempt was made to represent only the external building walls. A wall in ifcOWL can be defined as being internal or external using the `internalOrExternalBoundary_IfcRelSpaceBoundary` predicate. However, the ifcOWL test data used in this project varied, meaning that some buildings used this predicate in the definition of a wall whereas others did not. Furthermore, buildings had incorrect wall definitions of this predicate. The ifcOWL building `smallhouse_saref.ttl` had all walls defined as being external, even though some of the walls are clearly internal walls. Figure 4.5 below shows the external wall geometry of the ifcOWL building `AC20-FZK-Haus.ttl`, a building that had a correct external definitions of its walls.

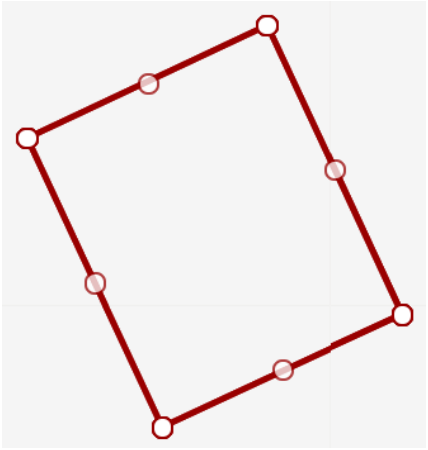


Figure 4.5: *External wall WKT geometry extracted from `AC20-FZK-Haus.ttl`.*

The usability of the results provided by this solution therefore depend on the quality of the input ifcOWL files. The geometry in Figure 4.5 above could be used in a 2D GIS system, matching how 2D GIS geometries are represented in Figure 4.4 above. However, more work is needed on the remaining WKT geometries presented in Table 4.2 above to give an external wall WKT representation of the buildings without use of the `internalOrExternalBoundary_IfcRelSpaceBoundary` predicate. An agreement could be make with the owner of a BIM model that requires them to adequately define the nature of building components, whether they are internal or external, before using the WKT building representation solution provided by this project. This would allow for the discovery and representation of external walls.

4.2.1.2 Runtime Performance of the Solution

An evaluation of the performance of this solution assists in determining the usability of the solution. The performance can be evaluated by examining the program runtime. A short program runtime is more desirable, allowing users to extract a 2D GIS geometry from their 3D BIM model in a timely manner. The program runtime was recorded for each of the six ifcOWL files given in Table 4.1 above. A comparison of runtimes can be performed based on the time taken to calculate the Local Placement (LP) of an `IfcWallStandardCase`, in milliseconds, and the time taken to calculate the Relative Placement (RP) of an `IfcWallStandardCase`, in milliseconds. The LP and RP runtimes are listed in Table 4.4 below. The triple count of each ifcOWL file is also given in Table 4.4, indicating the complexity of the ifcOWL building representation. It can be seen that as the complexity of the file in terms of triple count increases, so too does the time take to determine placement.

File Name	Triple Count	Triple Diff.	LP (ms)	RP (ms)
`smallhouse_saref.tll`	100246	0	776	1849
`Duplex_A_20110907_optimized.ttl`	205141	+ 104895	1945	4576
`Simple3-storeytestbuilding.ttl`	222545	+ 17404	2400	5642
`20170804_Musterhaus_MIT.ttl`	239899	+ 17354	2619	5850
`AC20-FZK Haus.ttl`	275685	+ 35786	2991	9900
`Barcelona_Pavilion.ttl`	1033905	+ 758220	13600	32400

Table 4.4: *Comparison of program runtime in terms of local and relative placement determination of an IfcWallStandardCase building entity.*

Table 4.4 shows that as the triple count of an ifcOWL file increases the time taken to extract the local and relative placement of a wall also increases. This is partly due to the time it takes Apache

Jena to search through an RDF model of an ifcOWL building. The bigger the RDF model, in terms of volume of RDF triples, the longer Jena takes to search and extract triples. The biggest file, `Barcelona_Pavilion.ttl`, takes over 13 seconds to determine the local placement of a wall and over 32 seconds to determine the relative placement of the same wall. This building contains 35 `IfcWallStandardCase` walls, therefore totalling approximately 26 minutes to extract all 35 walls! This is not particularly desirable for the owner of a 3D BIM model looking represent their model as a 2D GIS geometry. Future work on this project would focus on optimizing the Java code used to extract the 2D building representation, decreasing runtime to improve ease of use of the solution.

4.2.1.3 Limitations

An indication of the limitations of this solution has been given in subsections 4.2.1.1 and 4.2.1.2 above. The external walls should be represented only, matching the current 2D GIS standard. This external wall building WKT geometry is extracted for one of the test buildings but not for the others. The runtime of the program could also potentially limit the use of this solution. BIM model owners would not particularly want to wait long periods of time to extract a 2D WKT geometry, especially if a number of BIM models are to be processed. Another limitation of this solution is the fact that it only extracts ifcOWL walls that are of the type `IfcWallStandardCase`. There is no functionality to extract ifcOWL walls of the type `IfcWall`. However, future work could implement a similar solution to extract these types of walls as WKT. Finally, it has to be noted that the quality of the input ifcOWL test data can limit the solution results. For example, in `smallhouse_saref.tll` all walls are labelled as external, even though there are internal walls. This meant that a 2D WKT external wall geometry could not be extracted from the file.

4.2.2 Evaluation of ifcOWL Building Footprint Representation as 3D OBJ

The solution to create a 3D OBJ model of an ifcOWL building aims to connect BIM and 3D GIS. This solution can be seen as an extension of the process to extract a 2D WKT geometry, where the 2D wall layout is used to determine the positions of walls. This solution is evaluated in terms of the accuracy and usability of results and the limitations associated with the solution.

4.2.2.1 Accuracy and Usability of Results

As discussed in Section 4.2.1.1, the accuracy of the results can be examined visually, comparing the BIM IFC model with the resulting 3D OBJ model in Table 4.3 above. Examination of Table 4.3

shows that the 3D OBJ models do generally capture the wall structure of the parent IFC models. The OBJ walls have the correct placement and dimensions. However, it is noticeable that there are gaps at the corners of some walls, where the corner points of walls do not correctly align with the adjacent wall. This is due to a plane with zero thickness being used to represent the walls. All walls in the BIM IFC models have a thickness value, representing the real world thickness of the wall. This is something that has been excluded from the 3D OBJ representation of the building. Inspection of the 3D OBJ walls in `Simple3-storeytestbuilding.ttl` and `20170804_Musterhaus_MIT.ttl` show gaps and disconnections between wall endpoints as a result of the zero thickness plane. However, in `AC20-FZK-Haus.ttl` and `smallhouse_saref.tll` there are no such gaps at the wall endpoints. The placement of the walls in a 3D OBJ model is determined by the transformation of the walls' polyline, as performed in the solution to extract a 2D WKT geometry. It would seem that in `AC20-FZK-Haus.ttl` and `smallhouse_saref.tll`, the polyline is being defined at the external face of each wall rather than the internal face. The OBJ wall plane is therefore drawn at the external face of each wall for these ifcOWL buildings, avoiding the issues seen as a result of the zero thickness planes and giving a more accurate OBJ representation of the building.

Another noticeable element in the OBJ results is that some walls are missing entirely from certain models. Examination of the 3D OBJ model for `20170804_Musterhaus_MIT.ttl` shows this. Most of the left hand wall is missing along with some internal walls. This is due to this ifcOWL file in particular having wall types of both `IfcWallStandardCase` and `IfcWall`. As previously discussed, the process to extract a 2D geometry does not extract walls of the type `IfcWall`. This 2D geometry forms the basis of the 3D OBJ model and therefore walls of the type `IfcWall` are excluded from the OBJ model. A solution to extract and represent `IfcWall` walls as a 2D geometry would allow for the 3D OBJ representation of these walls.

The results obtained from this solution are useful in determining the 3D wall structure of a BIM building. 3D representation of a building's external walls would be useful in representing a 3D GIS version of the building as seen in a 3D GIS system. The roof of the building could easily be provided as another plane to complete the 3D building footprint outline. Although, a 3D OBJ model of the internal walls of a BIM building is also useful. Mapping tools such as Google Maps facilitate the viewing of a building's indoor wall structure for large public buildings such as shopping centres. These 3D OBJ models, representing both the internal and external walls, could be used in such a context, to allow a user to view the internals of a building in a 3D GIS application.

4.2.2.2 Limitations

As discussed above, one of the limitations with this solution is the exclusion of wall thickness. In some cases this can result in gaps between adjacent walls, lowering the accuracy of the OBJ building representation. This solution also assumes that there are no building components between walls on different building storeys. The OBJ building representation for the file `Simple3-storeytestbuilding.ttl` in Table 4.3 above shows clear gaps between walls on different building storey levels. This is due to the existence of a thin slab placed between storeys. Figure 4.6 below shows the BIM model for `Simple3-storeytestbuilding.ttl`. There is a slab between the ground floor walls and the first floor walls and a slab between the first floor walls and the second floor walls. This solution only models the walls of a building as 3D OBJ, excluding other entities such as slabs.

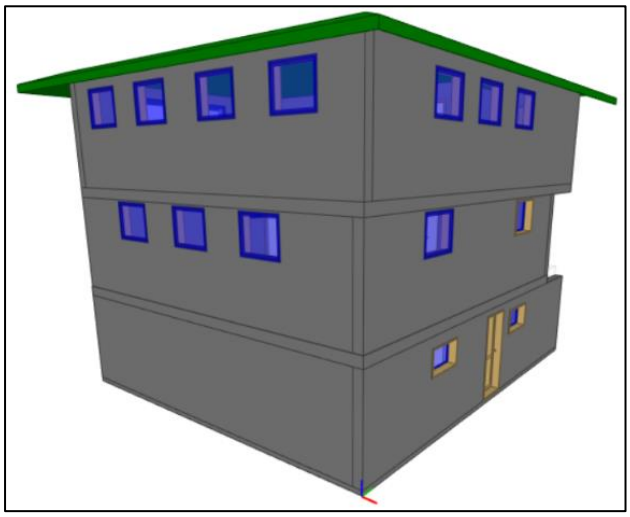


Figure 4.6: *Screenshot of `Simple3-storeytestbuilding.ttl` as viewed in BIM Vision. Slabs can be seen between building storeys, resulting in gaps in the OBJ model.*

4.2.3 Evaluation of ifcOWL to Building Topology Ontology (BOT)

The motivation behind creating a BOT representation of an ifcOWL building is to provide a more accessible and easy to use Linked Data BIM building representation with added 2D GIS component. This solution is not therefore evaluated on accuracy of geometries, but is evaluated on the usability of the resulting BOT building.

4.2.3.1 Usability of Results

A BOT building representation is created for each input ifcOWL building, representing the building using a finite number of BOT building components. The resulting BOT files from this solution contain a BOT Site, BOT Building, BOT Storeys, BOT Spaces and BOT Elements, with each component connected through a relationship hierarchy. Furthermore, a 2D GIS aspect is added to the BOT files, 2D WKT geometries, facilitating the representation of the building in a 2D GIS context.

Appendix B gives a snippet of the resulting BOT file from the `Simple3-storeytestbuilding.ttl` input ifcOWL file. The BOT triples in Appendix B contain definitions of BOT building components, definitions of the *has* predicate to connect building components and GeoSPARQL predicate definitions to represent 2D geometries as WKT. There are no further complexities added to the file. The user of a BOT building, resulting from this solution, does not need to spend a large amount of time learning the schema to work with the file. The opposite is true when working with ifcOWL building representations, where the complexities associated with ifcOWL would require a user to spend time learning the schema before commencing work on such a file.

The accessibility of a BOT file can therefore be measured by comparing the RDF triple count of a BOT building with its corresponding ifcOWL building. Table 4.5 below gives the BOT and ifcOWL RDF triple counts for the six test building files used in this project.

File Name	BOT Triple Count	ifcOWL Triple Count
`smallhouse_saref.tll`	47	100246
`Duplex_A_20110907_optimized.ttl`	305	205141
`Simple3-storeytestbuilding.ttl`	132	222545
`20170804_Musterhaus_MIT.ttl`	45	239899
`AC20-FZK Haus.ttl`	75	275685
`Barcelona_Pavilion.ttl`	115	1033905

Table 4.5: *Comparison of RDF triple count for a building represented as BOT and ifcOWL. Note, the BOT triple counts do not increase sequentially as with the ifcOWL triple counts. This is primarily due to the number of walls defined in the building.*

As can be seen in Table 4.5 above, there is a stark difference in RDF triple count between the resulting BOT and ifcOWL building representations. An ifcOWL building is a RDF version of a BIM IFC building model, capturing all physical and non-physical building components given in the IFC model. A large number of RDF triples is therefore needed to represent all this information. BOT, on the other hand, aims to represent a building's topological components, requiring only a small

number of RDF triples. A user examining their building in a 2D GIS context is more likely to prefer working with the BOT building representation due to its simplicity and accessibility. The generation of a BOT building with added 2D WKT geometries removes the need for a user to spend time learning the ifcOWL schema. It can be concluded that the BOT building representations resulting from this solution do achieve the solution goal of providing a more usable and accessible version of an ifcOWL building.

4.2.3.2 Limitations

The primary limitation associated with this solution is that is relies on the quality of the input ifcOWL data. BOT building component relationships can only be created if they exist in the input ifcOWL file. Certain input ifcOWL files were missing definitions relating to how components are related to one another. The file `20170804_Musterhaus_MIT.ttl` for example was missing definitions to relate an `IfcSpace` to an `IfcWallStandardCase`, (which can be seen in the BOT triple count for this file in Table 4.5 above where it is lower than similar sized ifcOWL files). This meant that the BOT relationship hierarchy was broken at this point for this file, resulting in no BOT Space to BOT Element relationships. The owner of a BIM model must define these relationships in ifcOWL for the resulting BOT building representation to contain a complete building component relationship hierarchy.

4.2.4 Evaluation of GeoSPARQL Building Footprint Alignment

An evaluation of the solution to align building footprint geometries using GeoSPARQL can be performed by examining the results returned from the query. The YASGUI query interface is used to view query results in a 2D GIS context, allowing the verification of the solution. This solution is evaluated by examining the results produced in YASGUI and by exploring the limitations associated with the solution.

4.2.4.1 Accuracy of Results

An issue faced when creating this solution was the lack of test data. This makes it difficult to fully evaluate the solution. Ideally, the 2D WKT GIS geometry of a processed ifcOWL or BOT building would be matched with a corresponding WKT geometry representing the same physical building defined in a different domain. As mentioned in Section 4.1.4 above, an ifcOWL building and its corresponding BOT building (with an altered building geometry) were used as the test data in this solution. Furthermore, only one input ifcOWL building, `AC20-FZK Haus.ttl`, had an

external wall geometry representation, leading to this file being the only test ifcOWL building used in the solution. Querying of the ifcOWL and altered BOT representation of `AC20-FZK Haus.ttl` via YASGUI allows for examination of the resulting aligned buildings. Figure 4.7 below shows the overlapping ifcOWL and BOT WKT geometries for this file. The underlying red geometry is the ifcOWL WKT building geometry and the overlying green geometry is the BOT WKT building geometry.

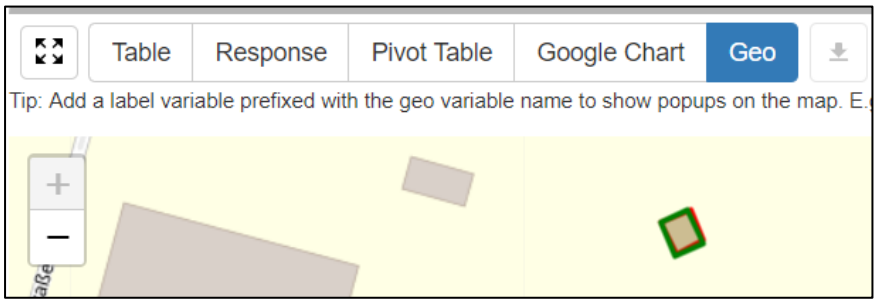


Figure 4.7: *Overlapping ifcOWL and BOT building geometries for `AC20-FZK Haus.ttl`, as viewed in the YASGUI Geo results page.*

It can be seen from Figure 4.7 that the geometries do align, as they meet the query filter conditions. Unfortunately this was the only ifcOWL file that could be tested with this solution. A full evaluation of this solution would require more ifcOWL test data. However, the solution was tested with building data sourced from Ordnance Survey Ireland (OSi). This building data did not contain any BIM information, although it did contain WKT building geometries that could be queried through GeoSPARQL. Appendix C contains the RDF triples of one of these buildings. As with above, for each building, a corresponding building had to be created containing an altered geometry. The solution works well for these OSi buildings, matching their geometries based on the GeoSPARQL `sfOverlaps` function and the geometry area values. Figure 4.8 below gives overlapping WKT geometries of two OSi buildings drawn in the YASGUI *Geo* results page. The red geometry is the original building geometry whereas the green geometry is the altered version of the original geometry.

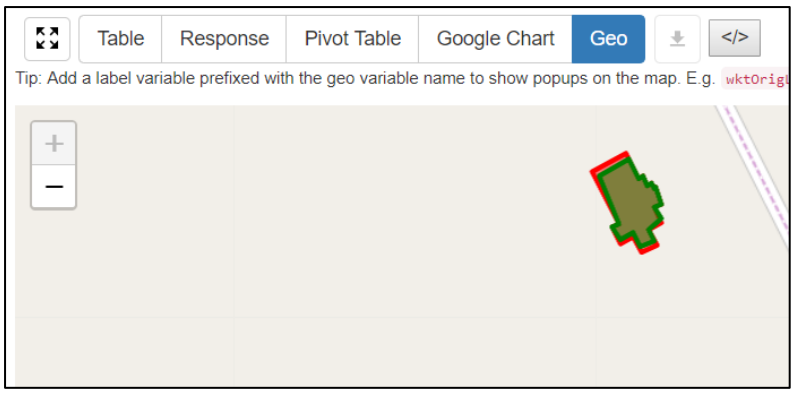


Figure 4.8: *Screenshot of the YASGUI Geo results page showing two overlapping OSi buildings.*

4.2.4.2 Limitations

A limitation of this solution is the need to calculate the area of a building's polygon geometry. The building must be processed to determine the polygon area before querying can be performed. A building must also be defined as a WKT `POLYGON` rather than a WKT `MULTILINESTRING`. The GeoSPARQL `sfOverlaps` function only checks fully closed overlapping polygons. This requires the conversion of a `MULTILINESTRING` to a `POLYGON` before the area can be determined. A process to convert these two WKT geometries is difficult to create as the polygon points must be in a sequential order. This conversion from WKT `MULTILINESTRING` to `POLYGON` was implemented for the external building geometry of the `AC20-FZK Haus.ttl` ifcOWL file, which has a simple four walled geometry. A similar process could not be implemented for the other five test ifcOWL files. A conversion process was not needed, however, for the OSi buildings, where the WKT geometries are given as WKT `POLYGON`. The usability of this solution therefore relies on how the WKT geometries are specified. A building with a geometry defined as a WKT `POLYGON` could easily be used with this solution, allowing the building to be aligned with the same building defined in a different domain. The same could not be easily said for a building geometry defined as a WKT `MULTILINESTRING`.

4.3 Project Challenges

The nature of this research project gave rise to a number of challenges. These challenges had to be overcome in order to progress with this project. A number of the challenges faced throughout the course of this project are discussed below in Sections 5.1.1, 5.1.2 and 5.1.3.

4.3.1 Learning New Technologies

The primary challenge that I faced when commencing work on this research project was that I had no previous experience with Linked Data or BIM technologies. The concept of Linked Data was entirely new to me before taking on this project. I was therefore required to learn about Linked Data theory and technologies. Aiding my learning was my participation in the module *Knowledge and Data Engineering (CS7IS1)* [50], conducted by Prof. Declan O'Sullivan of the School of Computer Science and Statistics at Trinity College Dublin. This module taught me the theory behind Linked Data, in particular about RDF, SPARQL, GeoSPARQL and the creation of ontologies. A significant amount of time was spent learning about Linked Data concepts and technologies during the first few months of this project. Once I was familiar with Linked Data and, in particular RDF, I could begin learning about BIM and ifcOWL.

Similar to my experience with Linked Data, I had no previous knowledge of BIM, IFC or ifcOWL before taking on this research project. As mentioned throughout this document, the ifcOWL schema is complex and comprehensive. Time must be spent learning how the ifcOWL schema models a building before interaction with ifcOWL files. It was a challenge to learn the layout of an ifcOWL building file and to discover how the RDF triples within the file relate to one another. Added to this was the challenge of learning how to use the Apache Jena framework to extract information from ifcOWL buildings. I was competent in Java programming before this project however I had never before encountered the Apache Jena framework. All these challenges of learning and working with new technologies had to be overcome before designing and implementing the solutions presented in the solutions toolkit.

4.3.2 Shortage and Consistency of Test Data

The lack of test ifcOWL data was a challenge that I faced when designing, implementing and testing the implemented solutions. As mentioned in Section 4.1 above, all test data used in this project was sourced from an online ifcOWL repository, located at [43]. This was the only available source that provided free ifcOWL files and their corresponding IFC models. BIM data is sensitive in nature, publicly releasing models of buildings could breach the privacy of the building owner. This could be the reason as to why BIM data is scarce on the internet. The test data chosen for this project, listed in Table 4.1 above, was selected as the files were comprehensive and complete building models, representing walls using the type `IfcWallStandardCase`. A number of files on the online ifcOWL repository were incomplete building models, contained structures that were not buildings or did not represent walls as `IfcWallStandardCase` (this project only

extracts walls of this form). Only a small number of files available on the repository satisfied these requirements and were therefore suitable for use in this project.

An added challenge relating to the testing of ifcOWL files was the consistency of data. Difficulties arose during this project as a result of different ifcOWL ontology versions being used to model the buildings. Sections 3.3.3.2 and 3.4.3.2 discuss the issues relating to ontology versioning, in terms of extracting the height of a wall and defining how objects relate to one another to generate the BOT relationship hierarchy. Each ontology version required examination of how that version models certain relationships. This is reflected in the code, where the ifcOWL ontology version must be selected. Checks are also implemented at points throughout the code, following different conditional branches based on the ontology version.

4.3.3 Geometry Skewing Issues

The issue of 2D WKT geometry skewing was observed during the development of this project. The comparison of an output 2D WKT building geometry with its 3D IFC geometry showed incorrect wall dimensions in the 2D WKT geometry. The walls were being skewed in a north – south direction as a result of the conversion process from xy-coordinates to latitude and longitude coordinates. Figure 4.9 shows a skewed geometry on the left. The walls are elongated, not representing the correct wall dimensions. The right hand side of Figure 4.9 shows the top-down view of the IFC model that this WKT geometry represents.

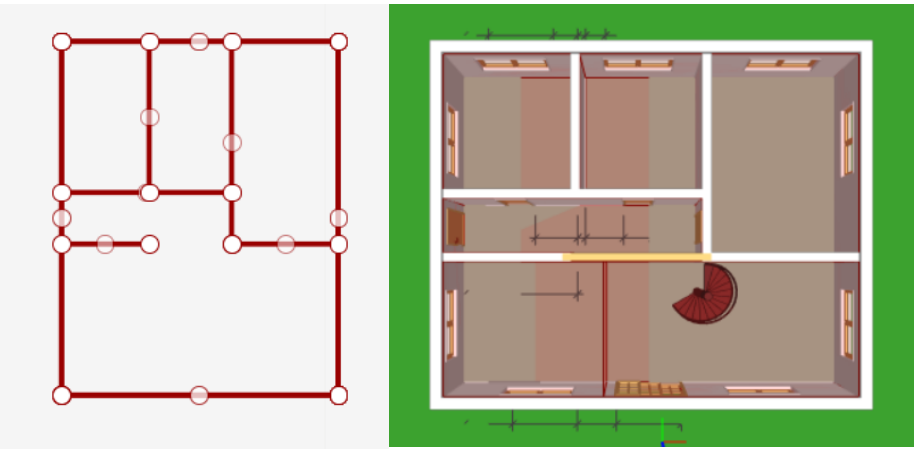


Figure 4.9: *The 2D WKT image on the left is the skewed geometry as a result of incorrect coordinate system placement of the walls. The image on the right is the top-down view of the parent IFC model, showing the correct wall dimensions for this building.*

Originally, the wall coordinates, extracted as metres, were converted to latitude and longitude values and then added to the latitude and longitude geolocation. This caused the geometry skewing. An alternative solution was implemented to overcome this skewing issue which first converted the geolocation to xy-coordinate metres before processing the building walls. The geolocation is converted from the WGS 84 EPSG:4326 projection (latitude and longitude coordinates) to the WGS 84 Web Mercator EPSG:3857 projection (xy-coordinates represented in metres). The extracted building walls are then added to the xy-coordinate geolocation value. All wall coordinates are then converted back to the EPSG:4326 projection, representing the building geometry as latitude and longitude coordinate values for use in the WKT geometry.

This was a challenge that took a significant amount of time to overcome. I exhausted a number of reasons as to why the skewing was occurring before finally realising it was the incorrect use of map projections. I had no further difficulties representing WKT geometries after solving this issue.

The challenges I faced throughout this research project gave rise to learning opportunities that I would have never previously anticipated. It is important to overcome these challenges to make progress in the project and to gain experience in how to solve potential future challenges relating to Linked Data BIM and GIS.

5. Conclusion

This research project demonstrates four methods that work towards the goal of integrating BIM and GIS buildings. The implemented solutions aim to address the issues in moving between the two domains by focusing on a building's geometric representation. BIM and GIS use different geometric representations and coordinate systems, making it a challenge to relate a BIM building to its corresponding GIS equivalent. A BIM building is therefore isolated, oblivious to its environmental surroundings. GIS data describing a BIM building can help enrich our understanding of a building and its surroundings. The methods presented by this research project facilitate the exploration of BIM in a GIS context, bridging the gap between the two domains.

The approach taken in this research project uses Linked Data technologies to integrate BIM and GIS buildings. A focus is placed on extracting a GIS representation from a BIM building defined in the ifcOWL format. The solutions toolkit contains two solutions that directly achieve this, with the other two solutions improving the accessibility of the BIM data. The solution to extract a 2D WKT geometry from an ifcOWL building integrates BIM with 2D GIS, facilitating the exploration of a BIM building in a 2D GIS context. The solution to extract a 3D OBJ model from an ifcOWL building integrates BIM with 3D GIS. This provides a method of exploring how a BIM building interacts with its external environment in a 3D GIS context. The complexity of IFC and ifcOWL building representations can be a deterrent for new users, unfamiliar with the technologies. The solution to create a BOT version of an ifcOWL building attempts to make a more accessible representation of a BIM building. This is achieved by providing a simple topological building representation as BOT. A geospatial element is added to this solution by attaching the 2D GIS geometries of the building components, providing another means of integrating BIM and GIS. The final implemented solution provides a means of interacting with integrated BIM data through GeoSPARQL querying. This facilitates the interlinking of BIM data with other domains.

The research objective set out by this project was to bridge the gap between GIS and BIM building representations through a Linked Data approach. The four implemented solutions provided by this project do achieve this goal. The results achieved show how GIS building representations can be extracted from Linked Data BIM building models, providing methods of both 2D GIS and 3D GIS integration. It is clear, however, that the methods provided are not a complete solution to BIM and GIS integration. Although they do provide initial exploration into the representation of BIM buildings as GIS, connecting the two domains.

5.1 Future Work

The work carried out during this research project provided some interesting insights into the challenges that exist in relating a BIM building to its corresponding GIS representation. The solutions toolkit demonstrates methods that help in overcoming these challenges by focusing on processes to extract a BIM building geometry for representation as 2D GIS and 3D GIS. However, further research could refine these solutions by providing even more usability and functionality for users.

The solution to extract and represent an ifcOWL building as a 2D WKT building footprint uses the building wall structure to create the geometry. The ifcOWL ontology allows for walls to be defined as both `IfcWallStandardCase` and `IfcWall`, representing walls that have a non-changing thickness and walls that have a changing thickness. This project only extracts walls from ifcOWL buildings that are of the type `IfcWallStandardCase`. Future work could extend this solution to extract both `IfcWallStandardCase` and `IfcWall` walls from an ifcOWL building. This would not limit the solution to only one type of wall. There is potential to further extend this solution in extracting other ifcOWL building components such as `IfcBeam`, `IfcChimney`, `IfcColumn`, `IfcCurtainWall`, `IfcDoor`, `IfcWindow` and `IfcRoof`. Extraction of these building components would make a significant improvement to the 3D OBJ models produced by this project, giving a richer definition of a building.

The usability of the solutions toolkit is important to the outcome of this project. The solutions aim to give BIM model owners methods of exploring their building in a GIS context by allowing them to extract building representations as GIS. Section 4.2.1.2 discusses the runtime performance of the solution to extract a 2D WKT geometry from an ifcOWL building file. The analysis of runtime shows that the solution can be slow for comprehensive BIM models with a large number of RDF triples. This is partly due to how Apache Jena extracts the information from ifcOWL files. My prior inexperience with this framework restricted me in finding the optimal solution to extract RDF data. Future work with this solution would optimise how Jena extracts information from ifcOWL files, improving the runtime of the solution and therefore increasing the usability of the solution.

One of the issues associated with BIM data is the scarcity of publically available IFC and ifcOWL files. This is primarily due to the sensitivity of the data and the complexities associated with developing IFC models, resulting in IFC developers being reluctant in releasing their models for free. The lack of usable test files available had an effect on this project. The robustness of the solution could not be measured as only a small number of ifcOWL files were suitable for testing. Future work on this research project would test the implemented solutions with a greater number of ifcOWL files. A more diverse and larger set of test files would aid in finding potential

downfalls of the implemented solutions, which may have been overlooked during this project as a result of the small number of ifcOWL files. This would improve the robustness and accuracy of the solutions, increasing the likelihood of these solutions developing into a standard process to integrate BIM and GIS.

5.2 LDAC Research Paper Submission

One of the contributions made by this project was the submission of a research paper to the 7th Linked Data in Architecture and Construction (LDAC) Workshop [51], which takes place in Lisbon between the 19th and 21st June 2019. This is an annual workshop that provides teachings in the use of Linked Data for the architecture and construction industry. The research paper, co-authored by Prof. Declan O'Sullivan and Dr. Kris McGlinn of the School of Computer Science and Statistics at Trinity College Dublin, details three of the four solutions presented in this project: the solution to extract a 2D WKT geometry from ifcOWL, the solution to represent an ifcOWL building as BOT and the GeoSPARQL building alignment solution. Acceptance of the paper into the workshop would indicate the significance of the work carried out by this research project and would show potential real world applications of the solutions provided by this project. Notification of acceptance for research papers submitted to the LDAC workshop is 22nd April 2019.

5.3 Concluding Remarks

The solutions presented by this research project do not solve the issues that exist in integrating BIM and GIS. Instead they provide an alternative approach of BIM and GIS building alignment through the extraction and representation of building geometries. This approach has potential in becoming a new standard of BIM and GIS integration, providing a starting point for developments in how to extract GIS representations from BIM buildings, connecting the domains. Although, there is much to be done in improving the accessibility of BIM data before a standard integration process can be defined. The complexities associated with BIM data can aid in repelling new developers from working with the schema, hindering research progress in the area. The use of Linked Data BIM building representations naturally improve the openness of BIM data, especially with the introduction of the Building Topology Ontology (BOT). Linked Data BIM could therefore provide the answer in connecting BIM and GIS buildings. This research project presents initial exploration into integrating Linked Data BIM and GIS building geometries, with the overall aim of finding that answer.

References

1. Bilal, M., Oyedele, L.O., Qadir, J., Munir, K., Ajayi, S.O., Akinade, O.O., Owolabi,H.A., Alaka, H.A., Pasha, M.: Big data in the construction industry: A review of present status, opportunities, and future trends. Advanced Engineering Informatics **30**(3), 500 – 521 (2016). https://doi.org/https://doi.org/10.1016/j.aei.2016.07.001, http://www.sciencedirect.com/science/article/pii/S1474034616301938
2. Bizer, C., Heath, T. and Berners-Lee, T., 2011. Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts* (pp. 205-227). IGI Global.
3. McGlinn, K., Debruyne, C., McNerney, L., O'Sullivan, D.: Integrating Ireland's Geospatial Information to Provide Authoritative Building Information Models. In: Proceedings of the 13th International Conference on Semantic Systems - Semantics2017. vol. 13, pp. 57–64. ACM Press (2017). https://doi.org/10.1145/3132218.3132223
4. McGlinn, K., Wagner, A., Pauwels, P., Bonsma, P., Kelly, P., O'Sullivan, D.: Interlinking Geospatial and Building Geometry with Existing and Developing Standards on the Web. Automation in Construction **tbd**, n/a (2019)
5. buildingSMART. 2013. *Industry Foundation Classes IFC4 Official Release*. [ONLINE] Available at: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/. [Accessed 7 March 2019].
6. World Wide Web Consortium, 2014. RDF 1.1 concepts and abstract syntax.
7. Pauwels, P. and Terkaj, W., 2016. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, *63*, pp.100-133.
8. Battle, R. and Kolas, D., 2011. Geosparql: enabling a geospatial semantic web. *Semantic Web Journal*, *3*(4), pp.355-370.
9. Rasmussen, M.H., Pauwels, P., Hviid, C.A. and Karlshøj, J., 2017. Proposing a central AEC ontology that allows for domain specific extensions. In *2017 Lean and Computing in Construction Congress*.
10. Lott, R., 2015. Geographic information-Well-known text representation of coordinate reference systems.
11. Paulbourke.net. (n.d.). *Object Files (.obj)*. [ONLINE] Available at: http://paulbourke.net/dataformats/obj/ [Accessed 12 March 2019].
12. Apache Jena. 2018. *A free and open source Java framework for building Semantic Web and Linked Data applications.* [ONLINE] Available at: https://jena.apache.org/. [Accessed 12 March 2019].
13. Liu, X., Wang, X., Wright, G., Cheng, J., Li, X. and Liu, R., 2017. A state-of-the-art review on the integration of Building Information Modelling (BIM) and Geographic Information System (GIS). *ISPRS International Journal of Geo-Information*, *6*(2), p.53.
14. Gröger, G. and Plümer, L., 2012. CityGML–Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *71*, pp.12-33.
15. Succar, B., 2009. Building information modelling framework: A research and delivery foundation for industry stakeholders. *Automation in construction*, *18*(3), pp.357-375.
16. Taylor, J.E. and Bernstein, P.G., 2009. Paradigm trajectories of building information modeling practice in project networks. *Journal of Management in Engineering*, *25*(2), pp.69-76.
17. Amirebrahimi, S., Rajabifard, A., Mendis, P. and Ngo, T., 2015. A data model for integrating GIS and BIM for assessment and 3D visualisation of flood damage to building. *Locate*, *15*(2015), pp.10-12.
18. OGC. The OGC Landinfra Conceptual Model; Open Geospatial Consortium Inc: Wayland, MA, USA, 2014.
19. Hwang, J.R., Kang, H.Y. and Choi, J.W., 2012, November. Development of an editor and a viewer for indoorgml. In *Proceedings of the Fourth ACM SIGSPATIAL International Workshop on Indoor Spatial Awareness* (pp. 37-40). ACM.
20. Geiger, A., Benner, J. and Haefele, K.H., 2015. Generalization of 3D IFC building models. In *3D Geoinformation science* (pp. 19-35). Springer, Cham.
21. Donkers, S., 2013. Automatic generation of CityGML LoD3 building models from IFC models.
22. Grau, B.C., Horrocks, I., Kazakov, Y. and Sattler, U., 2008. Modular reuse of ontologies: Theory and practice. *Journal of Artificial Intelligence Research*, *31*, pp.273-318.
23. McGlinn, K., Debruyne, C., McNerney, L. and O'Sullivan, D., 2017, September. Integrating Ireland's Geospatial Information to Provide Authoritative Building Information Models. In *Proceedings of the 13th International Conference on Semantic Systems* (pp. 57-64). ACM.
24. McGlinn, K., Debruyne, C., McNerney, L. and O'Sullivan, D., 2017. Integrating Building Information Models with Authoritative Irish Geospatial Information. In *International Semantic Web Conference (Posters, Demos & Industry Tracks)*.
25. Bizer, C., Heath, T. and Berners-Lee, T., 2008, April. Linked data: Principles and state of the art. In *World wide web conference* (pp. 1-40).

26. Open Geospatial Consortium. 2019. *OGC LandInfra / InfraGML*. [ONLINE] Available at: https://www.opengeospatial.org/standards/infragml [Accessed 24 Mar. 2019].
27. Open Geospatial Consortium. 2019. *IndoorGML, OGC Standard for Indoor Spatial Information*. [ONLINE] Available at: http://www.indoorgml.net/ [Accessed 24 Mar. 2019].
28. Safe Software. 2019. *FME – The Simple Solution for Complex Integration.* [ONLINE] Available at: http://www.safe.com/ [Accessed on 24 March 2019].
29. Ordnance Survey Ireland. 2019. *Osi*. [ONLINE] Available at: http://ontologies.geohive.ie/osi/index.html [Accessed 24 Mar. 2019].
30. buildingSMART. 2013. *Industry Foundation Classes IFC4 Official Release*. [ONLINE] Available at: http://www.buildingsmart-tech.org/ifc/IFC4/final/html/. [Accessed 24 March 2019].
31. buildingSMART. 2019. *buildingSMART - The Home of BIM*. [ONLINE] Available at: https://www.buildingsmart.org/ [Accessed 24 Mar. 2019].
32. Braden, B., 1986. The surveyor's area formula. *The College Mathematics Journal*, *17*(4), pp.326-337.
33. Rietveld, L. and Hoekstra, R., 2013, May. YASGUI: not just another SPARQL client. In *Extended Semantic Web Conference* (pp. 78-86). Springer, Berlin, Heidelberg.
34. ArcGIS Pro. (2019). [ONLINE] Available at: https://pro.arcgis.com/en/pro-app/ [Accessed 31 Mar. 2019].
35. Karan, E.P., Sivakumar, R., Irizarry, J. and Guhathakurta, S., 2013. Digital modeling of construction site terrain using remotely sensed data and geographic information systems analyses. *Journal of Construction Engineering and Management*, *140*(3), p.04013067.
36. Irizarry, J. and Karan, E.P., 2012. Optimizing location of tower cranes on construction sites through GIS and BIM integration. *Journal of information technology in construction (ITcon)*, *17*(23), pp.351-366.
37. buildingSMART. 2013. *IfcSite*. [ONLINE] Available at: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcproductextension/lexical/ifcsite.htm [Accessed 1 Apr. 2019].
38. buildingSMART. 2013. *IfcGeometricRepresentationContext* [ONLINE] Available at: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcrepresentationresource/lexical/ifcgeometricrepresentationcontext.htm. [Accessed 1 April 2019].
39. buildingSMART. 2013. *IfcWall.* [ONLINE] Available at: http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcsharedbldgelements/lexical/ifcwall.htm. [Accessed 1 April 2019].
40. buildingSMART. 2016. *ifcOWL ontology (IFC4_ADD1)* [ONLINE] Available at: http://ifcowl.openbimstandards.org/IFC4_ADD1/index.html [Accessed 4 Apr. 2019].
41. buildingSMART. 2016. *IFC Version* 4. [ONLINE] Available at: https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD1/HTML/ [Accessed 4 Apr. 2019].
42. buildingSMART. 2013. *IFC 2x Edition 3*. [ONLINE] Available at: https://standards.buildingsmart.org/IFC/RELEASE/IFC2x3/TC1/HTML/ [Accessed 4 Apr. 2019].
43. Ghent University. 2019. *IFC repository*. [ONLINE] Available at: http://smartlab1.elis.ugent.be:8889/IFC-repo/ [Accessed 4 Apr. 2019].
44. Foundation, B. 2019. *Home of the Blender project - Free and Open 3D Creation Software*. [ONLINE] Available at: https://www.blender.org/ [Accessed 4 Apr. 2019].
45. Datacomp. 2019. *BIM Vision | Free IFC Model Viewer*. [ONLINE] Available at: https://bimvision.eu/en/free-ifc-model-viewer/ [Accessed 6 Apr. 2019].
46. Wicket. 2019. [ONLINE] Available at: https://arthur-e.github.io/Wicket/sandbox-gmaps3.html [Accessed 6 Apr. 2019].
47. International Association of Oil & Gas Producers. 2019. *EPSG:3857*. [ONLINE] Available at: https://epsg.io/3857 [Accessed 6 Apr. 2019].
48. International Association of Oil & Gas Producers. 2019. *EPSG:4326*. [ONLINE] Available at: http://spatialreference.org/ref/epsg/wgs-84/ [Accessed 6 Apr. 2019]
49. BBN Technologies, Inc. 2009. *Parliament. A High-Performance Triple Store, SPARQL Endpoint, and Reasoner*. [ONLINE] Available at: http://parliament.semwebcentral.org/. [Accessed 6 Apr. 2019].
50. School of Computer Science and Statistics, Trinity College Dublin. 2019. *Knowledge and Data Engineering Module Descriptor*. [ONLINE] Available at: https://scss.tcd.ie/modules/?m=CS7IS1. [Accessed 7 April 2019].
51. *LDAC 2019*. 2019. 7th *Linked Data in Architecture and Construction Workshop*. [ONLINE] Available at: http://linkedbuildingdata.net/ldac2019/ [Accessed 9 Apr. 2019].
52. Zhang, C., Beetz, J.: Querying linked building data using SPARQL with functional extensions. In: Christodoulou, S.E., Scherer, R. (eds.) EWork and eBusiness in Architecture: ECPPM 2016 : Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM 2016), Limassol, Cyprus, 7-9 September 2016. pp. 27–34. CRC Press (2016), ISBN: 978-1-315-38689-8.

Appendix A1

```
inst:IfcWallStandardCase_67536

        ifcowl:name_IfcRoot  inst:IfcLabel_47874 ;

        ifcowl:objectPlacement_IfcProduct  inst:IfcLocalPlacement_67479 ;

        ifcowl:representation_IfcProduct  inst:IfcProductDefinitionShape_67531 .


inst:IfcProductDefinitionShape_67531

        ifcowl:representations_IfcProductRepresentation
inst:IfcRepresentation_List_47861 .


inst:IfcRepresentation_List_47861

        list:hasContents  inst:IfcShapeRepresentation_67514 ;

        list:hasNext      inst:IfcRepresentation_List_47862 .


inst:IfcRepresentation_List_47862

        list:hasContents  inst:IfcShapeRepresentation_67520 ;

        list:hasNext      inst:IfcRepresentation_List_47863 .


inst:IfcShapeRepresentation_67520

        rdf:type                          ifcowl:IfcShapeRepresentation ;

        ifcowl:contextOfItems_IfcRepresentation
inst:IfcGeometricRepresentationSubContext_375 ;

        ifcowl:representationIdentifier_IfcRepresentation  inst:IfcLabel_45055 ;

        ifcowl:representationType_IfcRepresentation  inst:IfcLabel_45062 ;

        ifcowl:items_IfcRepresentation  inst:IfcBoundingBox_67519 .


inst:IfcBoundingBox_67519

        rdf:type                       ifcowl:IfcBoundingBox ;

        ifcowl:corner_IfcBoundingBox  inst:IfcCartesianPoint_67517 ;

        ifcowl:xDim_IfcBoundingBox    inst:IfcLengthMeasure_44578 ;

        ifcowl:yDim_IfcBoundingBox    inst:IfcLengthMeasure_47344 ;

        ifcowl:zDim_IfcBoundingBox    inst:IfcLengthMeasure_47352 .


inst:IfcLengthMeasure_47352

        rdf:type           ifcowl:IfcLengthMeasure ;

        express:hasDouble  "3.5"^^xsd:double .
```

Appendix A2

```
inst:IfcWallStandardCase_932

        ifcowl:objectType_IfcObject  inst:IfcLabel_14561 ;

        ifcowl:objectPlacement_IfcProduct  inst:IfcLocalPlacement_905 ;

        ifcowl:representation_IfcProduct  inst:IfcProductDefinitionShape_930 .


inst:IfcProductDefinitionShape_930

        ifcowl:representations_IfcProductRepresentation
inst:IfcRepresentation_List_14557 .


inst:IfcRepresentation_List_14557

        list:hasContents  inst:IfcShapeRepresentation_910 ;

        list:hasNext      inst:IfcRepresentation_List_14558 .


inst:IfcRepresentation_List_14558

        list:hasContents  inst:IfcShapeRepresentation_928 .


inst:IfcShapeRepresentation_928

        rdf:type                        ifcowl:IfcShapeRepresentation ;

        ifcowl:contextOfItems_IfcRepresentation
inst:IfcGeometricRepresentationSubContext_88 ;

        ifcowl:representationIdentifier_IfcRepresentation  inst:IfcLabel_13936 ;

        ifcowl:representationType_IfcRepresentation  inst:IfcLabel_13967 ;

        ifcowl:items_IfcRepresentation  inst:IfcExtrudedAreaSolid_918 .


inst:IfcExtrudedAreaSolid_918

        rdf:type  ifcowl:IfcExtrudedAreaSolid ;

        ifcowl:sweptArea_IfcSweptAreaSolid  inst:IfcRectangleProfileDef_916 ;

        ifcowl:position_IfcSweptAreaSolid  inst:IfcAxis2Placement3D_31 ;

        ifcowl:extrudedDirection_IfcExtrudedAreaSolid  inst:IfcDirection_19 ;

        ifcowl:depth_IfcExtrudedAreaSolid  inst:IfcLengthMeasure_13953 .


inst:IfcLengthMeasure_13953

        rdf:type           ifcowl:IfcLengthMeasure ;

        express:hasDouble  "4000."^^xsd:double .
```

Appendix A3

```
inst:IfcWallStandardCase_2226
        ifcowl:name_IfcRoot          inst:IfcLabel_38186 ;
        ifcowl:objectType_IfcObject  inst:IfcLabel_38049 ;
        ifcowl:objectPlacement_IfcProduct  inst:IfcLocalPlacement_2181 ;
        ifcowl:representation_IfcProduct  inst:IfcProductDefinitionShape_2224 .

inst:IfcProductDefinitionShape_2224
        ifcowl:representations_IfcProductRepresentation
inst:IfcRepresentation_List_38183 .

inst:IfcRepresentation_List_38183
        list:hasContents  inst:IfcShapeRepresentation_2186 ;
        list:hasNext      inst:IfcRepresentation_List_38184 .

inst:IfcRepresentation_List_38184
        list:hasContents  inst:IfcShapeRepresentation_2222 .

inst:IfcShapeRepresentation_2222
        ifcowl:representationType_IfcRepresentation  inst:IfcLabel_38182 ;
        ifcowl:items_IfcRepresentation  inst:IfcBooleanClippingResult_2218 .

inst:IfcBooleanClippingResult_2218
        rdf:type  ifcowl:IfcBooleanClippingResult ;
        ifcowl:operator_IfcBooleanResult  ifcowl:DIFFERENCE ;
        ifcowl:firstOperand_IfcBooleanResult  inst:IfcExtrudedAreaSolid_2193 ;
        ifcowl:secondOperand_IfcBooleanResult  inst:IfcPolygonalBoundedHalfSpace_2217 .

inst:IfcExtrudedAreaSolid_2193
        rdf:type  ifcowl:IfcExtrudedAreaSolid ;
        ifcowl:sweptArea_IfcSweptAreaSolid  inst:IfcRectangleProfileDef_2191 ;
        ifcowl:position_IfcSweptAreaSolid  inst:IfcAxis2Placement3D_31 ;
        ifcowl:extrudedDirection_IfcExtrudedAreaSolid  inst:IfcDirection_19 ;
        ifcowl:depth_IfcExtrudedAreaSolid  inst:IfcLengthMeasure_37932 .

inst:IfcLengthMeasure_37932
        rdf:type           ifcowl:IfcLengthMeasure ;
```

Appendix B

```
<ifc_res/IfcSite_63972>
        a                   bot:Site ;
        geo:hasGeometry     <urn:geom:pt:3h7BG6MZ9Bjez2CwLfB_ey> ;
        bot:hasBuilding     <ifc_res/IfcBuilding_89> .

<ifc_res/IfcBuilding_89>
        a                   bot:Building ;
        geo:hasGeometry     <urn:geom:pt:3h7BG6MZ9Bjez2CwLfB_e$> ;
        bot:hasStorey       <ifc_res/IfcBuildingStorey_100>,
                            <ifc_res/IfcBuildingStorey_106> ,
                            <ifc_res/IfcBuildingStorey_112> ,
                            <ifc_res/IfcBuildingStorey_118> .

<ifc_res/IfcBuildingStorey_100>
        a                   bot:Storey ;
        bot:hasSpace        <ifc_res/IfcSpace_138> ,
                            <ifc_res/IfcSpace_573> ,
                            <ifc_res/IfcSpace_901> ,
                            <ifc_res/IfcSpace_1615> .

<ifc_res/IfcSpace_138>
        a                   bot:Space ;
        bot:hasElement      <ifc_res/IfcWallStandardCase_2306> ,
                            <ifc_res/IfcWallStandardCase_25593> ,
                            <ifc_res/IfcWallStandardCase_2389> ,
                            <ifc_res/IfcWallStandardCase_2226> .

<ifc_res/IfcWallStandardCase_2306>
        a                   bot:Element ;
        geo:hasGeometry     <urn:geom:pt:2S8SzRHDn6jvUE34AftCfh> .

<urn:geom:pt:3h7BG6MZ9Bjez2CwLfB_ey>
        geo:asWKT  "POINT (174.76095581027778 -36.85794448833333)"^^geo:wktLiteral .

<urn:geom:pt:3h7BG6MZ9Bjez2CwLfB_e$>
        geo:asWKT          "MULTILINESTRING     ((174.8303001104087     -36.87802668738343,
174.84965588618581    -36.87028480865067)    ,    (174.8233982376845    -36.826572992700235,
174.8571442975699     -36.88057821810389)    ,    (174.84579919362747    -36.86411388261699,
174.82546317515843    -36.872248472754514)    ,    (174.82232157767865    -36.865100621453614,
174.83143301384771    -36.86145573121415)    ,    (174.79049582929622    -36.90554272840519,
174.75759342090797    -36.85290530647895)    ,    (174.85630064607278    -36.87922855252449,
174.8233982376845    -36.826572992700235)    ,    (174.80351216668063    -36.83621958114997,
174.8300349627953    -36.87866297128478)    ,    (174.75873837651125    -36.85321069526211,
174.82301658581673    -36.82748947089245)    ,    (174.75843707240512    -36.85425543713594,
174.79133948079337    -36.90689192878252)    ,    (174.79322765319174    -36.840254709772175,
174.82536675784448    -36.89167991901221)    ,    (174.75885889815368    -36.85324284137885,
174.82297641193594    -36.827585941642525)    ,    (174.79045565541543    -36.90563910065197,
174.7574728992655    -36.852873160220284)    ,    (174.85587882032422    -36.880240803945284,
174.7900740035477    -36.90655463092504)    ,    (174.86805150621143    -36.875452575681216,
174.79003382966687    -36.90665100189362)    ,    (174.83585214073744    -36.867562591782644,
174.81828008526855    -36.83944127069658)    ,    (174.79176130654193    -36.90588003073636,
174.86877463606612    -36.87508300196035)    ,    (174.81367615852704    -36.8320716971873,
174.8363020882026    -36.868282526339016)    ,    (174.83656723581598    -36.867646155963094,
174.84579919362747    -36.8639531747594)    ,    (174.80359251444222    -36.83618742787576,
174.82572832277185    -36.87161213541454)    ,    (174.82297641193594    -36.827585941642525,
174.7571715951594 -36.853917906707345) )"^^geo:wktLiteral .

<urn:geom:pt:2S8SzRHDn6jvUE34AftCfh>
        geo:asWKT          "LINESTRING      (174.79045565541543      -36.90563910065197,
174.7574728992655 -36.852873160220284) "^^geo:wktLiteral .
```

Appendix C

```
<http://data.geohive.ie/resource/building/4aea987241d9461d9eccc5b705aca5ea>
       a      <http://ontologies.geohive.ie/osi#Building>,
              <https://w3id.org/bot#Building>,
              <http://www.opengis.net/ont/geosparql#Feature>;
       <http://www.w3.org/2000/01/rdf-schema#label>
              "not stated"@ga , "not stated"@en ;
       <http://ontologies.geohive.ie/geoff#hasForm>
              <http://ontologies.geohive.ie/osi/building#form112> ;
       <http://ontologies.geohive.ie/geoff#hasFunction>
              <http://ontologies.geohive.ie/osi/building#function376> ;
       <http://ontologies.geohive.ie/osi#lastUpdate>
              "2015-08-10T00:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;
       <http://www.opengis.net/ont/geosparql#hasArea>
              "377.5498046875"^^<http://www.w3.org/2001/XMLSchema#double> ;
       <http://www.opengis.net/ont/geosparql#hasGeometry>
              <urn:osi:build:geom:poly:4aea987241d9461d9eccc5b705aca5ea2013-12-
28T00:00:00> .

<urn:osi:build:geom:poly:4aea987241d9461d9eccc5b705aca5ea2013-12-28T00:00:00>
       a      <http://www.opengis.net/ont/geosparql#Geometry> ;
       <http://www.opengis.net/ont/geosparql#asWKT>
              "POLYGON     ((-8.62779469559618     52.6110130420158,     -8.62784571330338
52.6110256516979,       -8.62786930331218       52.6110314858333,       -8.62785977854575
52.6110458634856,       -8.62794658135117       52.6110673165767,       -8.62804366742607
52.6109210045195, -8.62795648343572 52.6108997242957, -8.6278975444235 52.6109888040467,
-8.62782274406145                  52.6109703073389,                   -8.62779469559618
52.6110130420158))"^^<http://www.opengis.net/ont/geosparql#wktLiteral> .
```