The Pennsylvania State University

The Graduate School

College of Information Sciences and Technology

INFORMATION INTEROPERABILITY BETWEEN BUILDING INFORMATION MODELING AUTHORING TOOLS AND SIMULATION TOOLS TO SUPPORT ENERGY EFFICIENT BUILDING DESIGN

A Thesis in

Information Sciences and Technology

by

Nan Yu

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2014

ABSTRACT

Effective Energy Efficient Buildings (EEB) design requires the use of Building Information Modeling (BIM) design authoring tools, along with various simulation tools to support decision making for optimized building solutions. This requires frequent interactions between computational tools. Traditionally, people have been using a point-to-point model for data exchange between those tools, which is complex and inefficient. An integrated data-centric model can reduce the communication cost and improve the interoperability. By setting up a BIM data hub in the center, both BIM design authoring tools and simulation tools only need to talk with the data hub. In this way, the communication interface among those tools is improved.

However, even in this model, each tool still requires an interface to connect to the BIM data hub. If part of those tools have the same kind of interface, and part of them are supported by one single tool, the data exchange model could be further simplified. In this case, the building lifecycle is divided into two modes: design and simulation. The lack of a unified interface to support information exchange and interoperability among different building design and simulation tools has become a bottleneck of the EEB design process. Therefore, a link between design mode and the simulation mode is required for the whole simulation process.

In this thesis, two existing infrastructures are leveraged to build a connected workflow using this simplified approach. The open source BIMserver is used as the information retrieval center, and OpenStudio is used as the information exchange and simulation platform. BIMserver can support the storage, maintenance, and query of Industry Foundation Classes (IFC) based building information models, and OpenStudio is a platform supporting whole building energy-related modeling and simulations. The main contribution of this thesis is to build an information exchange bridge between BIMserver and OpenStudio, which enables different design authoring tools and simulation tools that are connected to either of them to interoperate and exchange needed data.

This thesis describes the integrated approach at the data level, connecting BIMserver and OpenStudio to build a unified EEB data exchange model. The challenges of the seamless integration due to the dependency on both BIMserver and OpenStudio are also discussed in the thesis. The system, which organizes the data flow in a unified model, enabling effective exchange of data, has been open source released.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to first express my deep gratitude to my advisor Dr. Dinghao Wu for his kind guidance throughout my master program. Dr. Wu is an extremely nice advisor. I am impressed by his industrious and scrupulous attitude towards research. He has been very helpful on my research and life at Penn State. It is of great fun to work with Dr. Wu and learn a lot from him. I have grown much because of his influence.

I would also like to thank my other two committee members, Dr. John Yen and Dr. John I. Messner. We have been working on the same project for two years. I benefited a lot from their valuable advice and thoughtful comments. I appreciate their serving on my M.S. degree committee and guiding me in my research.

I am deeply grateful to my project team mates, Abdou K. Jallow, Sanghoon Lee, and Fangxiao Liu, for all their thoughtful feedback and help with everything. I have had a lot of fun to work with them. Also, I want to acknowledge my lab mates in 324, Lannan Luo, Xiao Liu, Jiang Ming, Yufei Jiang, Mo Yu, and others, who are all very nice and make me feel like we are a big and warm family. I would also like to thank those friends outside my research group, Bo Zhang, Chuangang Ren, Yu Pu, Xinyang Ge, Li Qiu, and Yibo Wu, who make my life at Penn State much more colorful and enjoyable.

Finally, I would like to give my special thanks to my beloved husband Yi Yang for his support, encourage, and love.

Chapter 1

Introduction

This chapter introduces the concept of Energy Efficient Buildings (EEB) and the critical phases in the EEB project. After an overview of the current situation in the AEC/FM Industry, the chapter describes the research objectives and strategy. The outline of the thesis is discussed at the end of this chapter.

Introduction to Energy Efficient Buildings (EEB) Project

Recently, the U.S. and rest of the world have devoted more attention to reducing energy consumption when new buildings are constructed (Foley, 2012). In the Architecture, Engineering, Construction and Facilities Management (AEC/FM) Industry, energy efficient buildings design is becoming more critical, especially as it relates to energy retrofit projects. In the process of energy efficient building design, decision-making in the very early stages can significantly influence the energy consumption (Pollock et al., 2009). The decision-making process should be built upon a channel, which connects the computational representation of a building's energy elements and the corresponding economic considerations (Jones et al., 2010). Energy modeling is such a channel providing designers with an outlook of potential energy consumption of varieties of designs prior to constructing the building (Fleming et al., 2012) to eliminate arbitrary decisions from the simulation process (Bazjanac, 2009).

During the building design lifecycle, EEB design depends on the collaboration of project participants using a variety of design tools and simulation tools to make decisions for the optimized building solutions. Building design authoring tools provide the data required by the simulation tools to conduct energy modeling and simulation. In this process, the data preparation for different simulation tools often reproduce already existing data created by design authoring tools (O'Donnell et al., 2013), which results in data fragmentation and inconsistency. Accordingly, seamless data exchange between building design authoring tools and simulation tools for building design, construction, and operation has been a goal of the AEC/FM Industry for decades (Hitchcock and Wong, 2011). Furthermore, different simulation tools running in different "energy simulation views" (Bazjanac, 2008) determine the varieties of data sets and data formats (Bazjanac & Kiviniemi, 2007) even in the simulation phase. It is quite necessary to agglomerate all energy simulation views into an integrated whole-building simulation methodology with the intent of exchanging data seamlessly (Guglielmetti et al., 2011). However, one of the most common shortcomings in the current industry practice is the lack of an integrated information exchange workflow. This causes fragmented connections and delays resulting in the inefficiency and ineffectiveness in the energy efficient building design process. Therefore, retrieval and exchange of building information in a timely and standard manner plays a major role in assuring efficient building energy simulation during the building design process. From the above discussion, we conclude that information retrieval and seamless information exchange are two core issues in the AEC/FM Industry for both the communication between building design authoring tools and simulation tools, and the communication amongst the simulation tools.

Research Objectives

It is difficult to retrieve 'knowledge' in the AEC/FM Industry (Redmond and Smith, 2011), and even harder to exchange data in different building lifecycle phases. To facilitate interoperability between design authoring tools and simulation tools for efficient information exchange, this research aims to provide an interoperable and integrated platform based on an open standards-based data format. The objectives include improving the software and data interoperability among the existing and new building design and simulation tools, and helping implement building design and simulation workflows using standards-based information exchanges. The final goal is to simplify, automate and integrate the information exchange processes among different tools, and develop enabling technologies and platforms to facilitate and establish an integrated eco-system around a BIM server with many tools and users.

Research Strategy

The strategy is to leverage existing infrastructure, where available, instead of starting from scratch. In the proposed workflow, Revit is used as the design modeling tool; Building Information Modeling (BIM) server platforms, such as 'BIMserver' implemented by bimserver.org (Beetz el al., 2010) as the information retrieval center; and OpenStudio as the information exchange and simulation platform. Revit is an application including features for architectural design, Mechanical-Electrical-Plumbing (MEP) and structural design using BIM (Autodesk, 2014). BIMserver supports the storage, maintenance, and query of Industry Foundation Classes (IFC) based building information models. OpenStudio, an interface to support whole building energy-related modeling and simulations, has another set of building energy modeling (BEM) representations (Weaver et al., 2012). Disconnect between different models prevents architects, engineers, and researchers from easily conducting integrated whole-building energy analysis (Guglielmetti et al., 2011). Accordingly, this thesis explores an integrated approach to leverage BIMserver and OpenStudio to enable open data exchange and interoperability among different building design and simulation tools. With the integrated approach, the inherent data inconsistency and mapping problems can also be solved.

Thesis Organization

The remaining of the thesis is organized as follows. Chapter 2 provides the background information about the concepts and representations in the AEC/FM Industry. Chapter 3 introduces several approaches to exchanging information among different simulation tools. Based on existing research results, this thesis proposes an integrated workflow to improve the software interoperability and building energy analyses efficiency. The validation tests are described in Chapter 4, followed by the limitations in Chapter 5. The last chapter summarizes the conclusion and discusses the future work.

Chapter 2

Background and Related Work

This research ventures into the complex realm of using BIM to support energy efficient building design. Data interoperability has long been a perplexing issue in the AEC/FM Industry. Tools and data formats are booming in their respective fields, such as the design phase and simulation phase, which makes the connection between BIM design authoring tools and simulation tools a tough mission. This chapter introduces the most commonly used tools and data representations in different views. At last, the chapter states the challenges of seamless information exchange between BIM design authoring tools and simulation tools.

BIM for Energy Efficient Buildings Design

The AEC/FM Industry shows increasing interest in adopting information technology in building designs (Bazjanac & Kiviniemi, 2007). In the U.S., there is around 5 billion square feet of new construction, 5 billion square feet of renovation, and 1.75 billion square feet of demolition every year. It is predicted that $400 billion will be saved annually if BIM is universally adopted (iwmsnews, 2009). BIM acts as a bridge between the AEC/FM Industry and information technology (Eastman et al., 2008), which makes the entire building lifecycle more efficient and effective, also leads to greater integration of AEC/FM stakeholders at the project design stage (Khalili and Chua, 2013). For effective exchange of information, the import and export of relevant information must be compatible with other tools (Bazjanac, 2007; O’Donnell et al., 2011). BIM is a shared digital representation of a building and its physical and functional characteristics, on the basis of open standards for software interoperability (O'Donnell et al., 2013). It serves as a process and an interoperable building model, which enables bi-directional data service for building design authoring tools and various simulation tools.

Current Simulation Tools for Building Energy Analysis

Building energy performance simulation leverages computer-based building energy analysis to quantitatively validate the correctness of decisions on building design and operations (Bazjanac et al., 2011). High-performance buildings require an integrated analysis, including whole-building energy, daylighting, airflow, among others. Traditionally, different simulation tools only concentrate on their respective domains. For instance, CONTAM is a multi-zone airflow and contaminant transport analysis software (Walton & Dols, 2010), EnergyPlus performs whole-building energy analysis, Radiance is used for daylighting and electric lighting simulation (Guglielmetti & Scheib, 2012), and Retrofit Manager Tool (RMT) is the most comprehensive building energy simulation web tool (Heidarinejad et al., 2014). Each of those tools in different simulation views requires the abstraction of the original data created by the design authoring tools in different granularities. Typically, designers adopt a point-to-point data exchange model (O’Donnell et al., 2011) between building design authoring tools and simulation tools, which is usually very complicated and inefficient. With the intent of facilitating information flow from an architectural design model to an energy model, Bazjanac and Kiviniemi (2007) developed a tool called the Geometry Simplification Tool (GST), superseded by Space Boundary Tool (SBT) (O'Donnell et al., 2013), which extracts a valid IFC geometric model and its construction properties, and transforms the original data into specific data structure and format required by EnergyPlus[1].

[1] http://apps1.eere.energy.gov/buildings/energyplus/

Building Information Representations

One of the essential barriers preventing seamless data exchange is the different building information representations. Each representation offers a range of proprietary file formats used in modeling and simulation tools. Revit (RVT) and Industry Foundation Classes (IFC) are two kinds of modeling representations in the design phase; EnergyPlus Input Data File (IDF), Green Building XML (gbXML), OpenStudio Model (OSM) and DXF are data representations and input file formats used in energy-related simulation tools. Though there exists so many proprietary data models of buildings, the IFC model, developed by buildingSMART who promote open BIM throughout the building lifecycle (buildingSMART, 2014a), is the only open specification that covers the entire lifecycle (O'Donnell et al., 2013). IFC provides EEB project participants and users of BIM with an open standard for sharing consistent, accurate building information amongst all computational tools used throughout a building's whole lifecycle (Hitch and Wong, 2011). It serves as a universal data exchange platform and eliminates the building model dependency from any specific tool (Khalili and Chua, 2013). Although IFC is the most mature open standard to represent building objects, it is only widely used in the building design phase. The simulation tools often adopt different data structures to represent different “views” of the building (IAI, 2006), which requires data format transformation, sometimes also content translation, of original data to form valid input for the simulation tools (Bazjanac, 2009). Due to the difference between the information representations, the transformation among them introduces human intervention, which also makes the process inefficient and ineffective.

Gap between BIM Design authoring Tools and Various Simulation Tools

BIM design authoring tools are used to create the building product design models, which describe the physical characteristics of building elements via their geometric and topological information (Khalili and Chua, 2013). EEB designers need enhanced methods to extract the geometry and topology from building design models to conduct various analyses (e.g. airflow analysis, energy analysis, and daylighting analysis). However, seamless information exchange between different BIM design authoring tools and various simulation tools has many challenges, such as heterogeneous requirements, different data formats and fragmented connections. The vast amount of diversity of project teams and stakeholders determines varying sets of needs of a BIM. This demands the provision of diverse independent tool functionality, features and operations, which makes it difficult to master them all. Most of the analysis and simulation software have been developed for different domains, and their formats are different in nature. Also, organizations tend to develop and use their internal standards and formats, which all limit model reuse, and information exchange and sharing among tools and stakeholders. Furthermore, specialized systems or application tools, which are often heterogeneous and disjoint, are required for individual domain specific activities, and include the use of information and models of varying scale and sources. The fragmentation creates integration and interoperability difficulties between the applications in the AEC/FM Industry (Liu et al, 2013). There are critical drivers for the widespread adoption of BIM in order to bridge the fragmentation which characterizes the industry and to facilitate effective management of information. Therefore, a highly integrated platform is of critical importance to facilitate interoperability and seamless information exchange.

Chapter 3

Approach

This research focused on building a bridge between BIM design authoring tools and simulation tools to facilitate seamless information exchange. In this thesis, Revit is used as the design modeling tool; BIMserver as the information retrieval center; and OpenStudio as the information exchange and simulation platform. The geometric information is very important in describing physical characteristics of building elements. A core challenge is the difference between a physical geometric representation of the object and an analytical geometric representation, e.g., the space boundary. Most of the simulation tools require enhanced methods to transform geometric representations from building design models. This research focuses on the extraction of geometry from IFC model and the transformation of the extracted data into the OpenStudio Model (OSM). In brief, the workflow consists of two basic tasks: geometric information extraction and geometric data transformation.

Before introducing the proposed approach, two other models are presented first and compared with the third simplified data exchange model.

The Traditional Point-to-Point Model

From the perspective of the Architecture, Engineering, Construction and Facilities Management (AEC/FM) Industry, the design of EEB is the panoramic motivation including not only the optimized solution of energy, but also other factors, such as daylighting, and airflow. As mentioned in Chapter 2, building information has various representations and simulation tools usually work in different simulation views. Therefore, direct data exchange between BIM design authoring tools and simulation tools, and the data exchange amongst simulation tools are required. Traditionally, people have been using the point-to-point interface for data exchange, as shown in Figure 3-1.

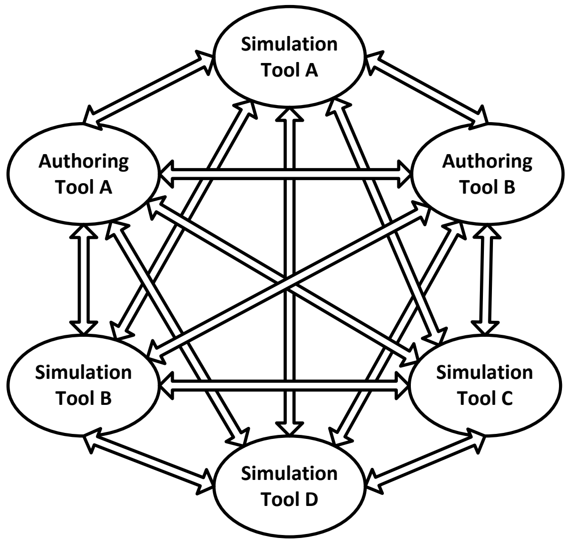

Figure 3-1. The Traditional Point-to-Point Model.

This point-to-point information exchange model is very simple and trivial for a small communication group. The complexity of this model increases dramatically as the number of simulation tools increases. As more factors are taken into account, more interactions will occur among those tools. In this approach, each pair needs to communicate once. Assuming this model has *n* nodes, after all nodes completing interactions with other nodes, it makes $O(n^2)$ interactions.

The Data-Centric Model

To reduce the communication cost, a data-centric model is illustrated in Figure 3-2. With a BIM data hub in the center, all computational tools that want to interact with others must communicate with the BIM data hub first, and then the BIM data hub transfers the interaction between two nodes. The advantage of the BIM data hub is even more obvious when the number of nodes increases. For the point-to-point method, each newly added node has to set up communication interfaces with all the existing nodes. For the BIM data hub, it only needs to set up one communication line for any newly added node and the interaction number is $O(n)$. In this way, the need for data modeling and service management at the individual tool level is reduced.

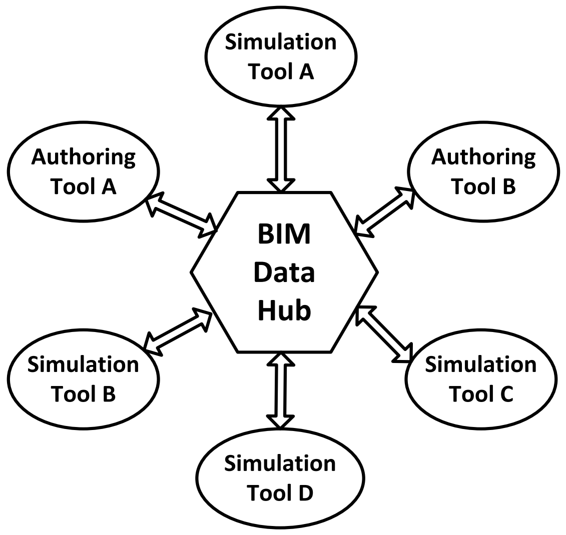

Figure 3-2. A Data-Centric Model.

The data-centric model has reduced the communication cost significantly; however, even in this model, each computational tool still requires one interface to connect to the BIM data hub. Whenever any interface changes, the BIM data hub has to adapt to the new interface. It is possible that BIM data hub becomes the bottleneck of the information exchange model. To leverage the existing building energy simulation eco-systems around OpenStudio, OpenStudio is added in the data-centric model. Some simulation tools, such as EnergyPlus and Radiance, are supported by OpenStudio. In this case, OpenStudio partakes in the data management of the BIM data hub. If Authoring Tool A and Authoring Tool B in Figure 3-2 have the same kind of interface, and the other simulation tools are supported by OpenStudio, this data-centric model could be further simplified. In practice, this is the common case. The computational tools are aggregated into two groups: design and simulation.

A Further Simplified Data Exchange Model

Figure 3-3 illustrates the simplified data exchange model. BIM data hub serves as a bridge between building design mode and simulation mode. In the design mode, BIM design authoring tools could be either Revit or ArchiCAD. In this thesis, Revit is selected as the BIM design authoring tool. The architectural designers create the building design model in Revit, and import the model into the BIM data hub in IFC format. The BIM data hub consists of two components: BIMserver and the transformation module. In the model, BIMserver is adopted as the implementation of a BIM server which takes charge of building information management and selective building information retrieval. It accepts the IFC format exported from BIM design authoring tools such as Revit or other tools that support IFC exporting, as input data. The input IFC data contains overall and original building geometric information and is loaded into the memory of BIMserver. In the advanced query mode, BIMserver executes the query code to selectively retrieve required building information. The transformation module links the BIMserver and OpenStudio, which enables different design and simulation tools to connect to either of them to interoperate and exchange needed data. The output of the BIM data hub is the transformed output file in OSM format. At last, OpenStudio reads the OSM file and perform supported simulation analyses. Currently, OpenStudio can support EnergyPlus for energy analysis and Radiance for daylighting analysis, and will support CONTAM for airflow analysis in the future.

According to Figure 3-3, the core work in this model includes using BIMserver for building information management and selective information retrieval, and using OpenStudio as a possible solution to perform various simulations. The remaining part states the implementation details about the two modules.

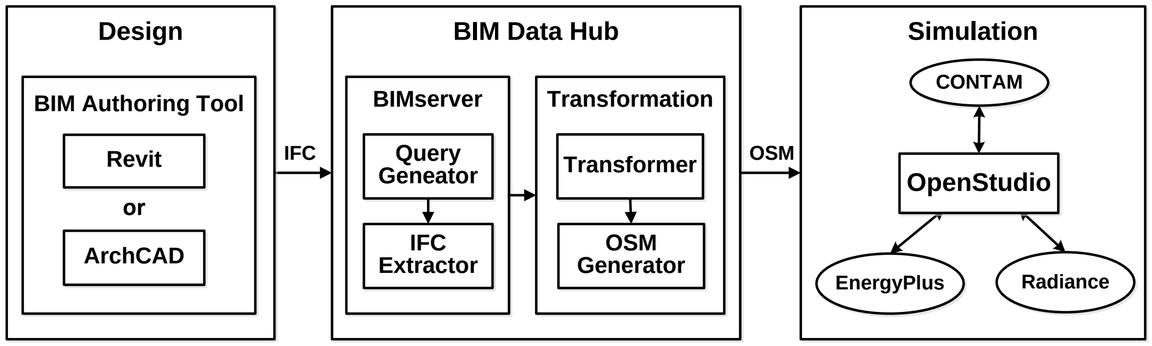

Figure 3-3. The Simplified Integrated EEB Data Exchange Model.

BIMserver for building information management and selective retrieval

As an application for building information management and information retrieval, BIMserver provides the users with a simple graphic user interface (GUI) to input Java query code manually. After successfully compiling and running, BIMserver retrieves the building information according to requirement specified in the query code. This query mechanism requires end-users to compose Java codes with deep knowledge of programming and IFC implementation, which creates a barrier that prevents the experts in the AEC/FM Industry from conducting information query unless they either learn programming skills or hire IT engineers with background knowledge in the AEC/FM Industry. It is desirable to generate the Java query code automatically for the BIMserver users, especially in the simplified data exchange model (Figure 3-3). BIM data hub accepts the IFC as the input data format, and exports OSM as the output data format. However, it is almost impossible for any server to extract or even interpret the geometric and topological information from the data design model implicitly or explicitly (Borrmann and Rank, 2009; Dominguez et al., 2011). The current practice of collecting building information from IFC and exporting the specific input required by the simulation tools is still manual transformation, which is not very efficient. The implicit and automatic extraction of geometric information from IFC has been a goal for decades in the AEC/FM Industry (Hitchcock and Wong, 2011).

On the basis of Model-View-Definition (MVD), Jiang et al. (2012) proposed that it is possible to automate the query process by generating queries from a MVD automatically. Following their work, a tool named *Query Generator* is developed to extract the geometric information of some building elements from IFC and transform the result into the format accepted by OpenStudio automatically. Through this automated process, human intervention in query generation is eliminated.

Query Generator

The Query Generator takes a MVD or an extended MVD as input, and then automatically generates queries that extract the needed information from IFC models. MVD is a subset of the IFC schema which is necessary to fulfill an information exchange requirement, and establishes the connection between domain description and IFC (buildingSMART, 2014b). The first step is to parse the input lexically. In this step, the whole content of the file is divided into meaningful segments. The second step is to parse it semantically to know what attributes are required by the MVD that users need to define. BIMserver maintains all the information. According to the sample codes given by BIMserver, all the codes contain a huge proportion of routine code and follow some patterns, which could be summarized and generalized. These features show the feasibility of generating the Java code automatically. The design fits the following primary principle: the Query Generator should finish the work automatically and in batch. In brief, the tool consists of three parts: the input parser, the intermediate representation generator, and the query exporter.

The input parser identifies the keywords that represent the IFC elements, and tokenizes those keywords. For each word, it invokes the IFC Extractor to extract the properties of the IFC elements, and store the data in the memory of BIMserver. The intermediate representation generator is actually composed by the Transformer and the OSM Generator in the transformation module. All the modules within BIM data hub are not isolated, but rather integrated together as a mixture. The last part is query exporter, which records the generated query code, saves it in the file system, and exports the generated query code into BIMserver. Eventually, the Java query code is inserted and run in the advanced query mode in the GUI of BIMserver, which searches the memory to extract required data and displays the query result in the console.

IFC Extractor

IFC Extractor is a sub-module in Query Generator. It keeps the IFC data structure for the original data and extracts the properties of building elements according to IFC specification. The extraction based on the IFC specification prevents arbitrary and manual data exchange, and preserves the integrity and standard of original data (Bazjanac, 2009).

BIMserver creates a Java class for each IFC element. The IFC specification states the relationship amongst the classes, such as inheritance, composition, and etc., as well as the properties and inverse properties of IFC elements. By means of the IFC Extractor, it is possible to extract building information in the level of elements or properties of elements. IFC Extractor also prepares extracted data for the following transformation.

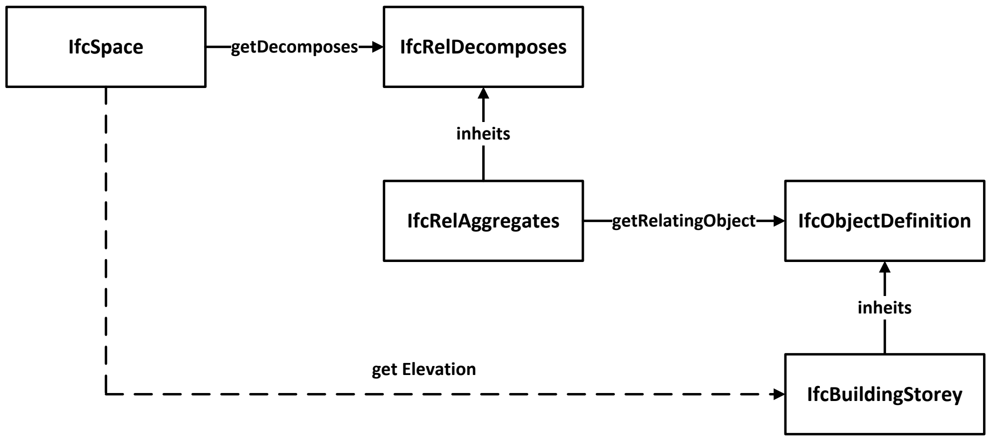

Figure 3-4. Space-Elevation Relation.

The geometry of building elements is extracted in this module. For IfcBuildingElement including IfcWallStandardCase (a derived subclass of IfcWall), IfcWindow, IfcDoor, and IfcSlab, the geometric information is represented in the 'ObjectPlacement' and 'Representation' properties. The property of 'ObjectPlacement' is used for translating different local coordinate systems; the property of 'Representation' is the boundary information of the elements. Besides these two properties, the relationship among the elements is also required. For example, the building storey elevation is needed for the extraction of the space boundary information. Figure 3-4 demonstrates a simple case about how to get the elevation from the IfcSpace.

As a preparation of transformation from IFC to OSM, the relationship between IfcSpace and IfcWall/IfcSlab, and the relationship between IfcWall and IfcWindow/IfcDoor have to be extracted in advance. The space to wall connection in IFC model is illustrated in Figure 3-5.

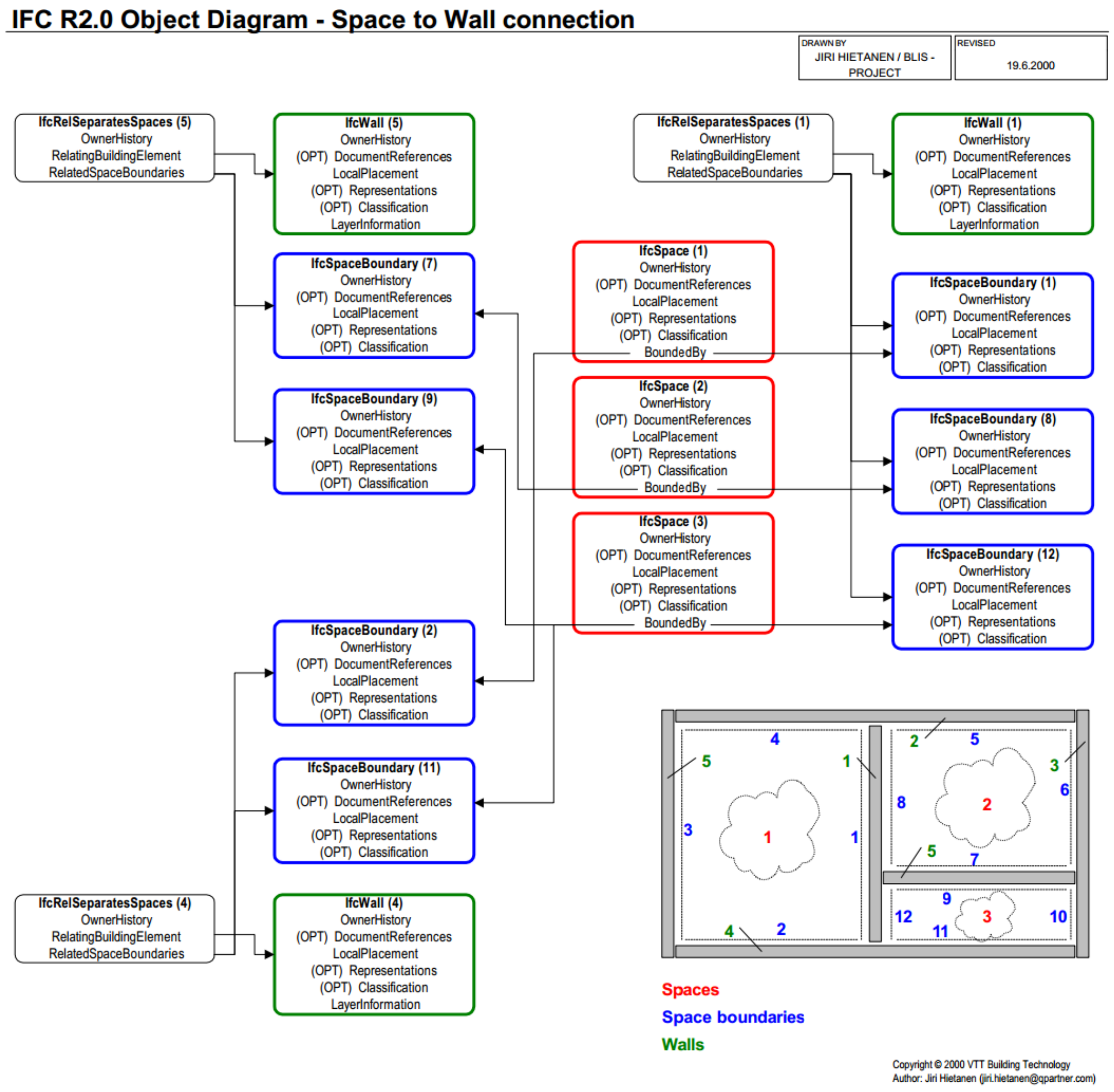

Figure 3-5. Space to Wall Connection (Hietanen, 2000). ©2000 VTT Building Technology. All rights reserved. Reproduced here for educational purposes only.

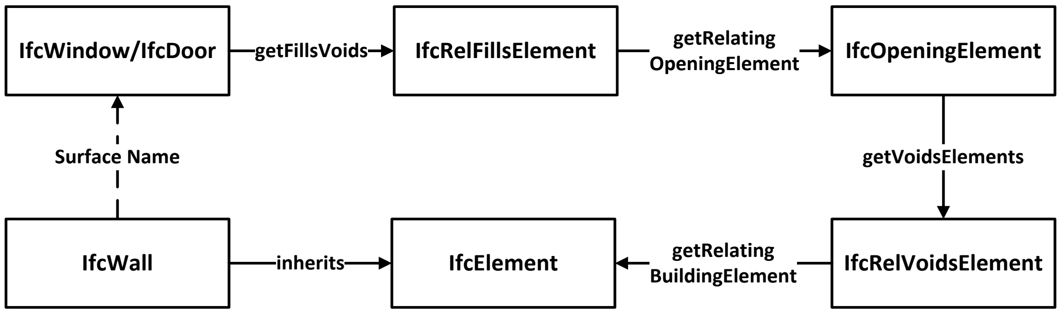

Figure 3-6. Wall-Window/Door Relation.

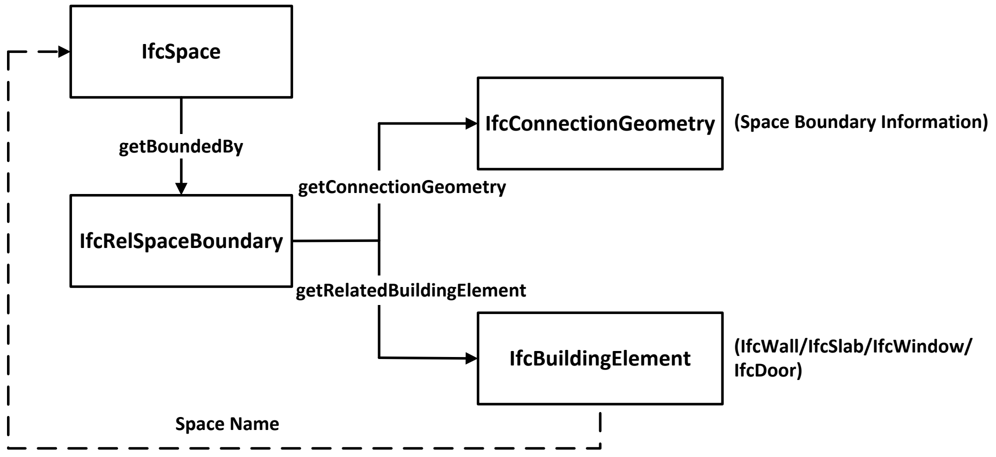

Figure 3-7. Space-Wall/Floor/Roof Relation.

Figure 3-6 and Figure 3-7 are two simplified version of Figure 3-5, which illustrate the relation link in IFC representation. The solid line shows the relation specified in IFC model, and the dashed line displays how the two elements are connected in OpenStudio model.

Integration through OpenStudio as a possible solution

BIMserver is an open-source vendor-independent module for information management and retrieval. Through the Query Generator and IFC Extractor, it extracts the building information according to the information exchange requirement in different granularities, such as elements level or properties level, and exports the retrieved result as plain text from its memory. The only issue is the repeated actions, such as parsing information exchange requirement and extracting building information, for any prospective simulation tools. The integration with OpenStudio solves this issue effectively. The required information is only extracted once, and OpenStudio processes the information for further analyses. The following section introduces the characteristics of OpenStudio which make it a possible solution as an information exchange and simulation platform, and the implementation details about other two sub-modules invoked by the Query Generator.

OpenStudio as an information exchange and simulation platform

The AEC/FM Industry experts and IT professionals are pursuing integrated models to reduce data remodeling and management for each simulation tool. Yu et al. (2013) propose that the integration of BIMserver and OpenStudio might be a potential solution for information exchange. Based on this assumption, the feasibility to choose OpenStudio as an interface platform is discussed.

OpenStudio is designed to establish an object-oriented framework for Building Energy Models (BEM), which is compatible with existing work (Guglielmetti, 2011). BIMserver is an object-oriented framework for building information and transforms all semi-structured information presented in IFC into building objects with corresponding attributes and relationships stored in memory. In view of the object-oriented characteristic, it is possible for data to flow from the BIMserver to OpenStudio and build an information exchange bridge for building simulations. Second, OpenStudio and EnergyPlus are bound together in the initial design (Ellis et al., 2008). Accordingly, EnergyPlus leverages OpenStudio to conduct whole-building energy analysis without extra effort. OpenStudio also supports Radiance to perform daylighting analysis besides conducting whole building energy analysis through EnergyPlus. OpenStudio is planning to support CONTAM to conduct airflow analysis in the near future. It is desirable to adopt a single tool to facilitate analyses from different perspectives.

OpenStudio is designed to overcome the shortage of user-friendliness of EnergyPlus and Radiance. Lack of an intuitive GUI for the tools creates a high entry barrier which prevents novel users from using the tools (Weaver et al., 2012). Other software applications such as DesignBuilder, IES, and eQuest try to present a state-of-the-art GUI to users. However, some of these software applications are commercial software, which are developed based on the internal standards. Users are constrained by the specialized GUI to make limited analyses. In contrast, OpenStudio is open-source, cross-platform and cross-language. Last but not least, OpenStudio provides a rapid development mode and open application programming interface (API), which makes it highly extensible and customizable. It is rather simple for developers to either build on existing applications or create completely new ones to perform customized building energy analysis (Weaver et al., 2012). All of these aspects suggest OpenStudio as a suitable platform for initial targeting to support the data exchange needs of building simulation modeling.

Transformer

According to the discussion in Chapter 2, there is a big gap between the IFC model and the OpenStudio model, not only the fragmented data formats, but also the fragmented connections and heterogeneous requirements. Therefore, the functionalities of the Transformer include both the format transformation and the content translation. Specifically, it transforms the format from IFC to OSM, translates different local coordinate systems for geometry extraction of building elements, transforms the geometry from the physical representation to the analytical representation, converts unit measurement, and approximates close boundary points of building elements.

(1) Format Transformation from IFC to OSM

Before transforming format from IFC to OSM, the corresponding relationship between them has to be figured out. Table 3-1 demonstrates the element mapping from RVT to IFC to OSM.

Table 3-1. Revit-IFC-OSM Element Mapping.

AutoDesk Revit (.rvt)	BIMserver (.ifc)	OpenStudio (.osm)	Note
Areas	IfcSpace	OS:Space	
Rooms			
Spaces			
Walls	IfcWall	OS:Surface	Surface Type: Wall
Roofs	IfcRoof		Surface Type: RoofCeiling
Floors	IfcSlab		Surface Type: Floor
Windows	IfcWindow	OS:SubSurface	Sub Surface Type: FixedWindow
Doors	IfcDoor		Sub Surface Type: Door
Site	IfcSite	--	Group 1
Levels	IfcBuildingStorey	--	Group 1
Stairs	IfcStair	--	Group 2
Railings	IfcRailing	--	Group 2
Ramps	IfcRamp	--	Group 2

This thesis focuses on the geometry extraction from IFC model, so the three most critical OSM elements are OSM:Space, OSM:Surface, and OSM:SubSurface. OSM:Space corresponds to IfcSpace in IFC model, which corresponds to three types of Revit elements: areas, rooms, and spaces. It is easy to find the space type of Revit element in IFC model through the property of 'Long Name' in IfcSpace. OSM:Surface corresponds to IfcWall, IfcRoof, and IfcSlab in IFC model, which corresponds to walls, roofs, and floors in Revit model respectively. The three kinds of surfaces are distinguished by the property of surface type in OSM. Particularly, most of the properties of flat roofs are extracted from IfcRoof, but the geometry of the flat roofs is extracted via the relation with IfcSlab. OSM:SubSurface corresponds to IfcWindow and IfcDoor in IFC model, which corresponds to windows and doors in Revit model respectively, and the types are distinguished by the property of sub surface type. The remaining elements occur in Revit model and IFC model, but not in OSM are divided into two groups. Group 1 contains the elements that are used to calculate the geometry of OSM:Space, OSM:Surface, or OSM:SubSurface. The elements in group 2 are omitted in OSM.

OSM requires the relationship between OSM:Space and OSM:Surface, and relationship between OSM:Surface and OSM:SubSurface to link the building elements together. Those relationships have already been extracted by the IFC Extractor module.

(2) Local Coordinate Systems Translation

IFC model uses relative coordinate systems to representation the geometry of building elements to remove the dependency amongst different layers, whereas OSM uses a unified global coordinate system. Therefore, the building elements in IFC model represented in their respective local coordinate systems need to be translated into the global system to keep the consistency in different models.

IFC model adopts the composition/decomposition to represent the relationship among the building elements. The aggregation relationship 'IfcRelAggregates' is a special type of the general composition/decomposition (or whole/part) relationship. Each 'IfcRelAggregates' relationship introduces a layer of relative coordinate system. Figure 3-8 denotes the aggregation relationship from the very top 'IfcProject' to a common element such as 'IfcWallStandardCase' of a very simple office building. In this figure, there are 4 times of local coordinate system translation (LCST). Each LCST is a 3D translation as shown in Figure 3-9. The 'ObjectPlacement' of IFC elements determines the translation for the coordinate systems. In most cases, the origin point of the coordinate system, the x axis, y axis, and z axis all change. The translation leverages the matrix translation to relocate the origin and the directions of the axes, and then calculates the vector components of each direction. The tricky part is that the IFC specification only specifies the x axis and z axis in the 3D coordinate system, which requires the calculation of y axis. The aggregation relationship starts from the top to the bottom, but the translation process should reverse from the bottom to the top.

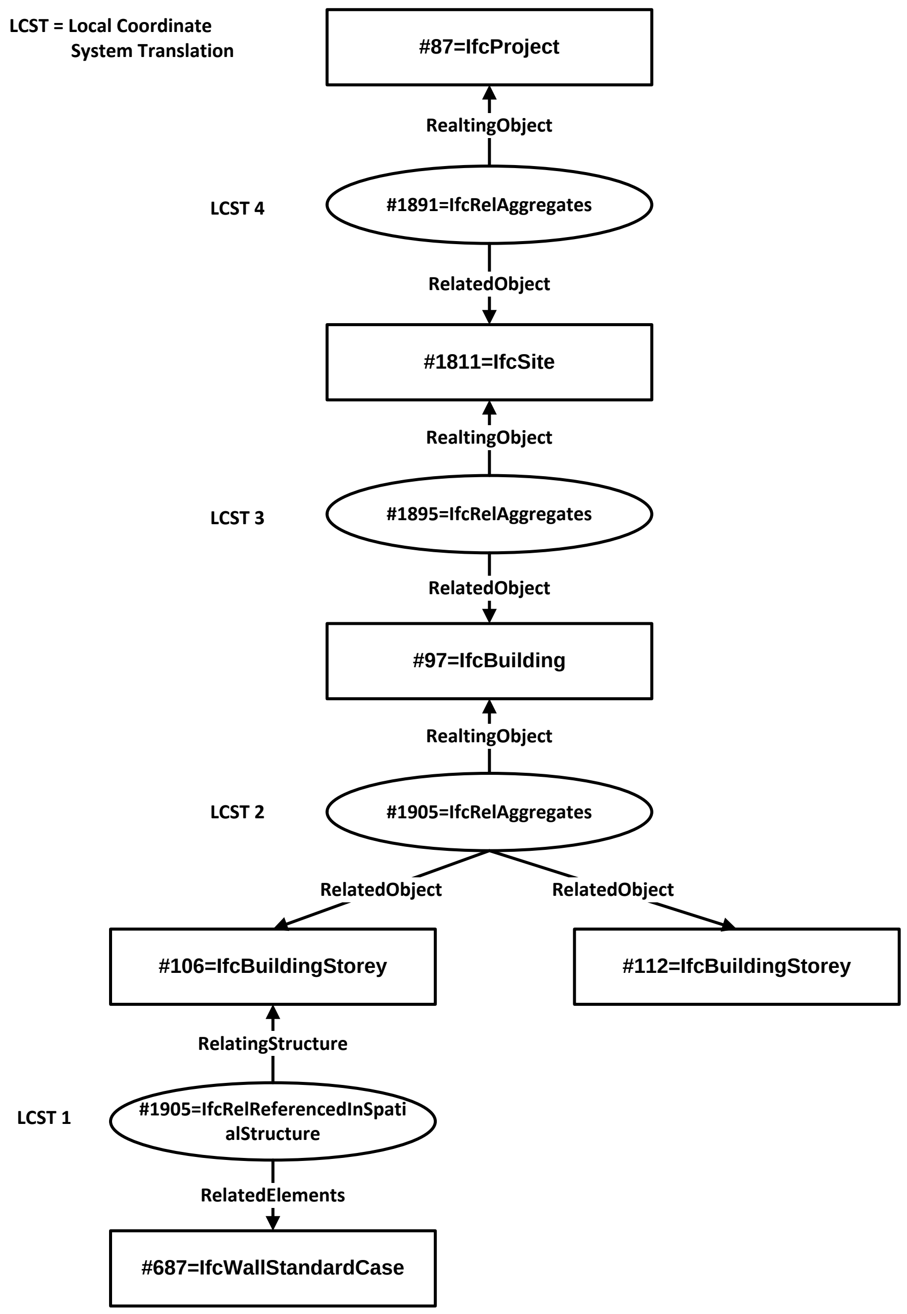

Figure 3-8. Aggregation Relationship amongst Building Elements.

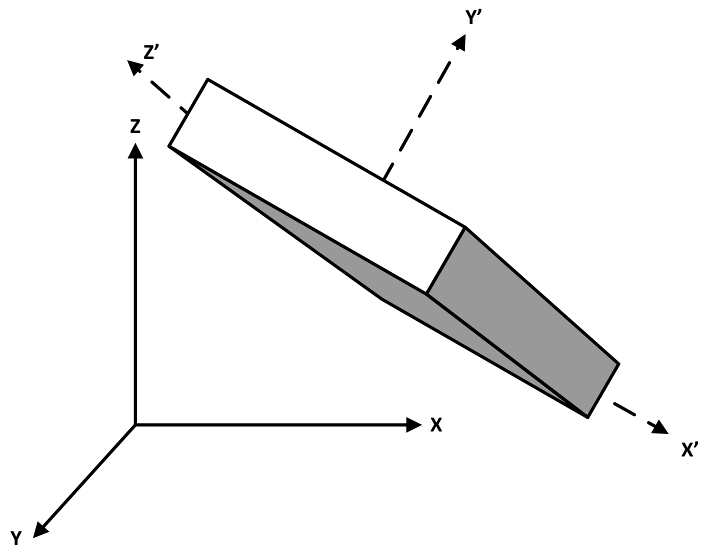

Figure 3-9. 3D Local Coordinate System Translation.

The translation above is the external 3D translation for a building element and its related elements. Some elements require an extra internal 2D translation. As mentioned in IFC Extractor, the geometry of building elements is represented in the property of 'Representation'. While within this element, some properties contain an internal relative coordinate system. For example, the 'IfcSweptAreaSolid' is a derived class for the geometric property of IfcSlab, which contains a property named 'Position' denoting the coordinate system for the swept area. After the internal 2D LCST and the external LCSTs, the geometry of IFC elements is translated into a unified global coordinate system.

(3) Geometric Representation Transformation

IFC model contains the physical geometry of building elements in detail, but energy analysis requires an analytical geometric representation which is expressed in a reduced and simplified data set. Therefore, the extracted geometry has to be transformed from the physical representation to the analytical representation before transmitting to OpenStudio. There are two approaches to transform the geometry: bottom-up and top-down. The bottom-up methods extracts the geometry of physical objects, and then the relation between surfaces and spaces to get the spaces. In the geometry extraction process, the thickness of materials is ignored for the geometry transformation. To avoid information loss, the thickness is stored along with the building objects in case inverse transformation is required. This thesis focuses on the information flow from building design tools to simulation tools. However, the inverse flow is also critical. It can be used to manage changes for energy efficient retrofit project (Liu et al., 2014).

To take a wall as an example, only four black points on the central axis of an IfcWallStandardCase are extracted, as shown in Figure 3-10. For IfcSlab, all points are in the same 2D plane. They are extracted and translated into 3D with the elevation of the building storey of the slab. The thickness of the slab is ignored. In OpenStudio, building designers often focus on the area ratio between walls and the attached windows/doors. Therefore, this thesis only extracts the peripheral points of IfcWindows and IfcDoors. The window sills, the frames and other complicated structures are all ignored.

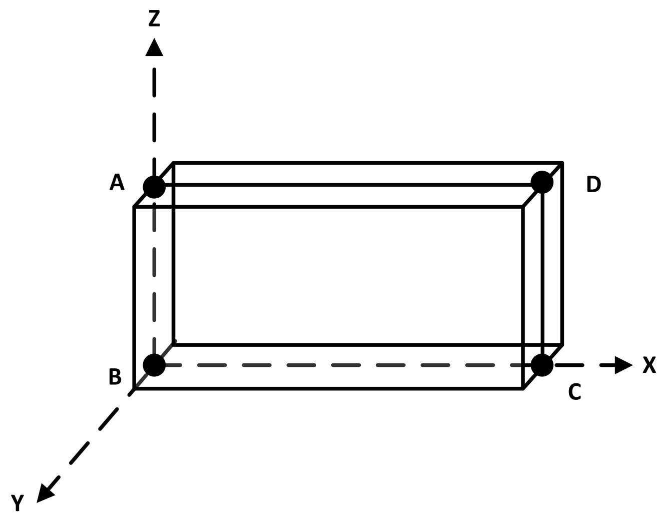

Figure 3-10. The Simplification of IfcWallStandardCase.

The top-down approach starts from spaces, along with the relationship between spaces and surfaces. For each attached surface, the space boundary geometry is extracted and transformed. The boundary information is the required analytical geometric representation for the building objects. According to the result, the top-down approach is more accurate.

(4) Units conversion

OpenStudio reads the geometry of OSM files in inch and foot by default, whereas IFC model uses the unit measurement according to its definition. There is a unit measurement convertor within the Transformer module. It analyzes the unit measurement adopted in the IFC model, and stores the conversion ratio in the Transformer's memory. After extracting all geometry points from IFC model, the convertor converts the units according to the conversion ratio.

(5) Close Points Approximation

The geometry transformation transforms the physical representation into an analytical representation, which results in gaps in the corners in Figure 3-11. To remove the gaps, a simple and naive approximation function is developed to group the close points and replace them with one point in the Transformer module. First, all the points on the surfaces and sub-surfaces are saved in a list, and sorted according to the x axis value, y axis value, and z axis value respectively. Then a reasonable threshold is set to represent the normal thickness of the building elements after several tentative experiments. As long as the distance between two points is less than the threshold, the two points are considered identical, and replaced using the smaller value. After the close points approximation, the gaps amongst building elements are eliminated in Figure 3-12.

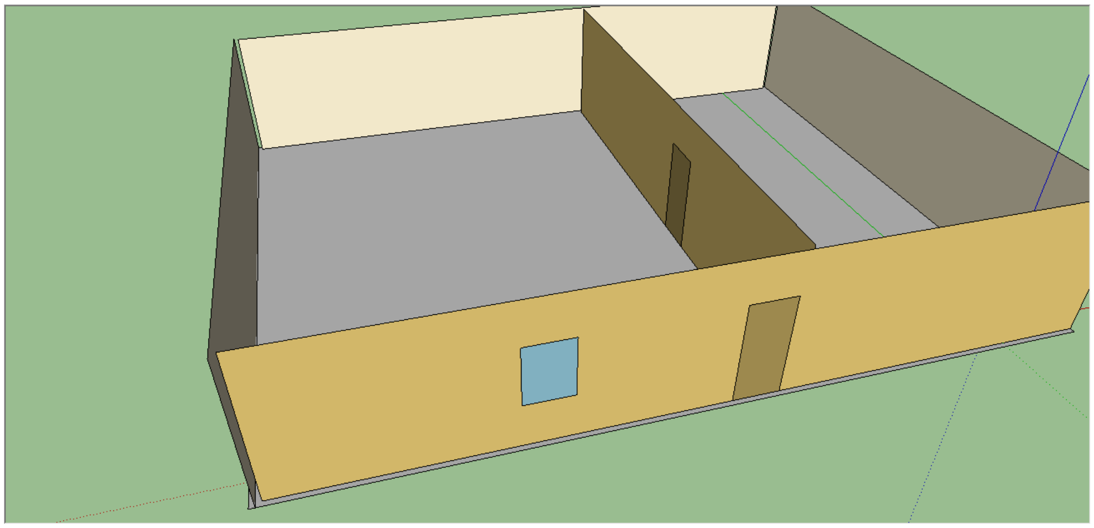

Figure 3-11. Gaps in Corners.

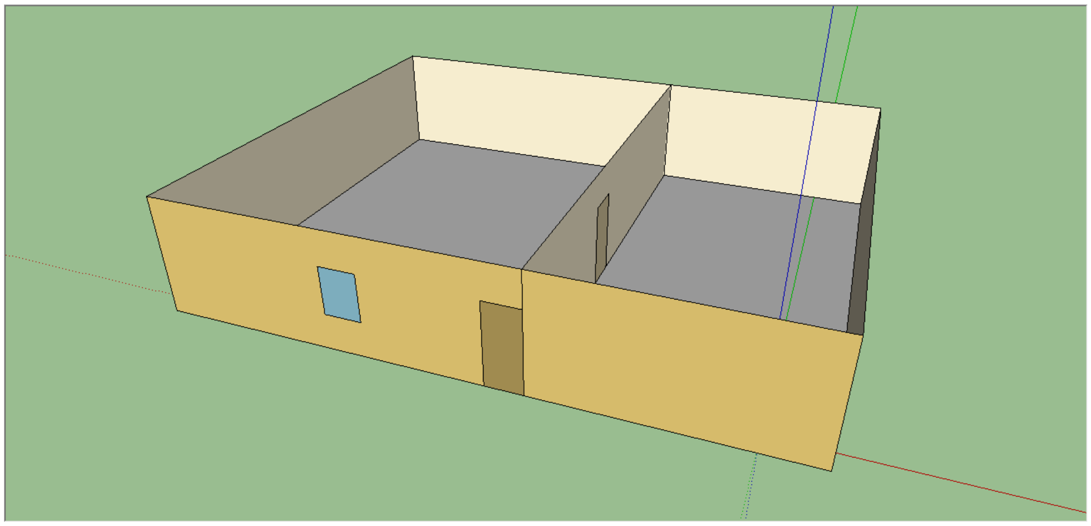

Figure 3-12. No Gaps in the Corners.

OSM Generator

Query Generator invokes the IFC Extractor to extract required IFC elements, and the Transformer module to transform both the format and content into a representation accepted by

OpenStudio. BIMserver executes the generated query code, and outputs the query result as plain text, which cannot be used by OpenStudio directly. Therefore, the OSM Generator module reorganizes the data structure of the query result, and generates an additional intermediate output in OSM. Figure 3-13 presents the components of a simple and standard OpenStudio model. The components in solid lines are the most important elements in OSM and have been implemented. Because designers often do not include the dashed components in the original IFC model, and energy engineers often focus on the geometry extraction first, those components in dashed lines are not covered in the current version.

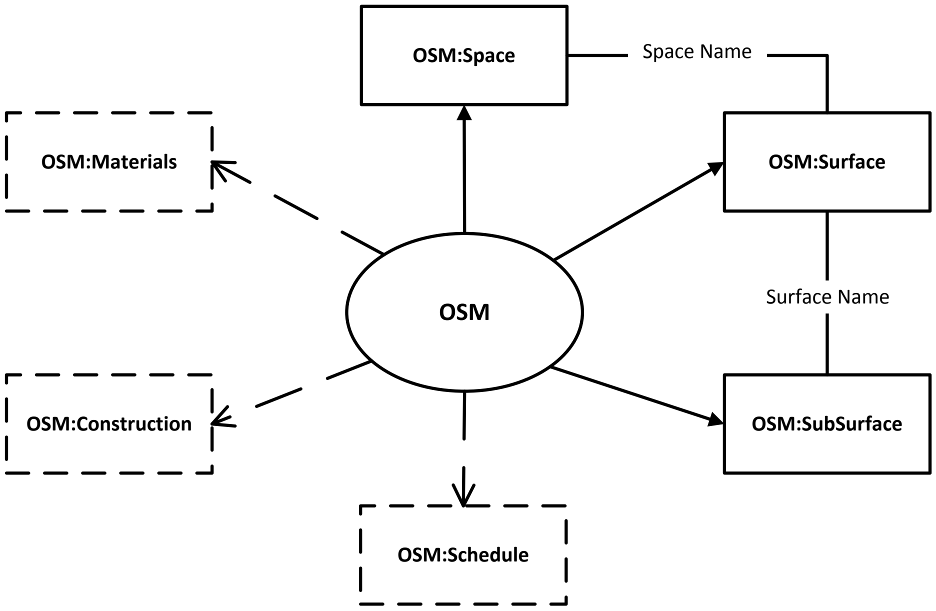

Figure 3-13. OSM Components.

There are three possible methods to generate the OSM objects from IFC model: (1) constructing OSM objects within a text file, and outputting the text file in '.osm' suffix, (2) re-creating the OSM objects through OpenStudio API, and (3) IFC to OSM through JSON. The first two methods are on the basis of a clear understanding of a standard OSM structure. In Method 1, three inner classes are constructed in OSM Generator to simulate the data structure of

OSM:Space, OSM:Surface, and OSM:SubSurface, along with an output generation function to generate the output file. They are demonstrated in code snippets of OSM Generator 1 in Appendix. In Method 2, the OpenStudio API (Application Programming Interface) is invoked to create the OSM objects. It avoids the potentially incorrect or nonstandard data structures of OSM objects, and builds the relationships between the objects such as OSM:Space and OSM:Surface according to the link specified in the program. The only concern for method 2 is the incompatibility between BIMserver and OpenStudio API. BIMserver is implemented in Java, while OpenStudio provides API in C#, Ruby, but no Java support currently. Therefore, for method 2, either a native programming interface or a separate module to run the OSM generator is needed. Both of the first two methods are customized for integration with OpenStudio. If BIMserver is going to integrate with other information exchange platforms, new customized output format generator is required.

The last method is accomplished via JSON format. JSON (JavaScript Object Notation) is a lightweight open data-interchange format (http://www.json.org). All the data which passes through BIMserver is object-oriented so that it can organize the data as building objects and export the query result in JSON format. The data in the JSON format needs to be parsed and re-organized into OSM format, which is the required and accepted input format by OpenStudio. This can be implemented by a programming script in either Ruby or C#. This method involves an additional transformation to JSON, which seems to be more complex. However, this method increases the flexibility of the data exchange model. The current exchange model leverages BIMserver and OpenStudio. If the BIM data hub plans to integrate with other information exchange platforms, which are similar as OpenStudio but requires different input formats, the unified intermediate transformation can avoid the embarrassment of the first two methods.

With the link between BIMserver and OpenStudio, the integrated model enables data exchange and the pipeline of whole-building simulation. The pipeline refers to a chain of building information processing stages. For example, data processed in a design model is exported from Revit into the IFC format and taken by BIMserver, which selectively transforms the needed data for OpenStudio. Other tools linked with OpenStudio can run whole-building simulations to perform various analyses. With the integrated data exchange model, it is possible to pipeline the data flow in a unified interface and therefore enable effective exchange of data.

Open-Source Release

The prototype implementation has been released with an open source license at Github (QueryGeneratorforBIMserver2.0, 2014). BIMserver users can download the application and extract the information from BIMserver for various analyses. The objective of this version is to integrate the BIM Datahub and OpenStudio to potentially facilitate more effective information exchange among different simulation tools and platforms. At present, this version supports query generation of data preparation for OpenStudio for energy related analysis. The application also defines several query templates for different Information Exchange Requirement (IER). It contains the following steps to run the application and conduct analysis using OpenStudio:

1. Users provide the Query Generator with the IER, which specifies the required data for BIMserver.

2. The application generates the query code automatically according to the IER.

3. Users open the generated query code, copy the content and paste it in the advanced query block of BIMserver.

4. BIMserver shows the queried result, and outputs the intermediate file for OpenStudio.

5. OpenStudio conduct energy related analysis.

Chapter 4

Validation Tests

To validate the accuracy and potential benefits of the simplified workflow, as well as the Query Generator presented in the thesis, three case studies are conducted. The first case is to examine the functionalities of the Query Generator by means of extracting information for CONTAM to conduct airflow analysis; the second case is to verify the accuracy of the extracted geometry of some building elements for a simple office building; the last case is to check the feasibility of the approach via a real building (Building 101) located in the Navy Yard in Philadelphia.

Case Study 1: Using CONTAM to Conduct Airflow Analysis

Before the simplified data exchange model proposed, some initial work to investigate information exchange between BIM design authoring tools and simulation tools was performed using airflow analysis (Jiang et al., 2014). It is the direct information extraction and exchange from an IFC model to CONTAM. Figure 4-1 shows an airflow analysis model development workflow using BIM. The horizontal level in the top part stands for the simulation process inside CONTAM, including zoning, specifying flow path, connecting HVAC (Heating, Ventilating, and Air Conditioning), to zones and running the simulation. It is of critical importance to build a tool to collect building information specified by the requirements and connect CONTAM with BIMserver. During the information collection process, a subset of building geometric information would be extracted via a query function according to MVD (model-view-definitions). In this case, four steps are adopted to collect the required building information. The first step is to design the information exchange requirement (IER) and abstract it to a standard MVD. The MVD defines the data set required by CONTAM. Step 2 is to generate the query code automatically on the basis of the MVD, followed by the execution of the query code. The last step is to translate the query result into JSON format that CONTAM understands.

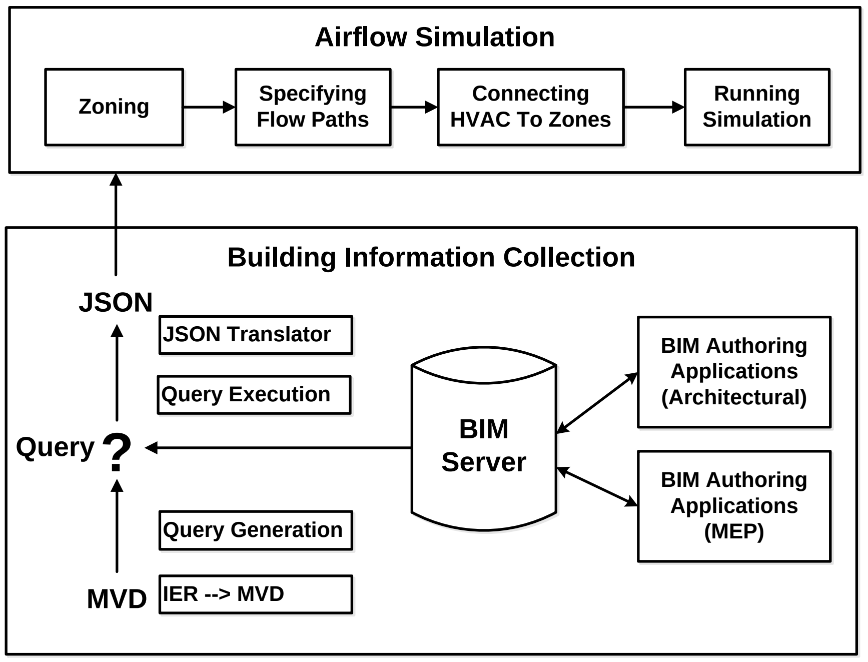

Figure 4-1. Airflow Analysis Model Development Using BIM (Jiang et al, 2014).

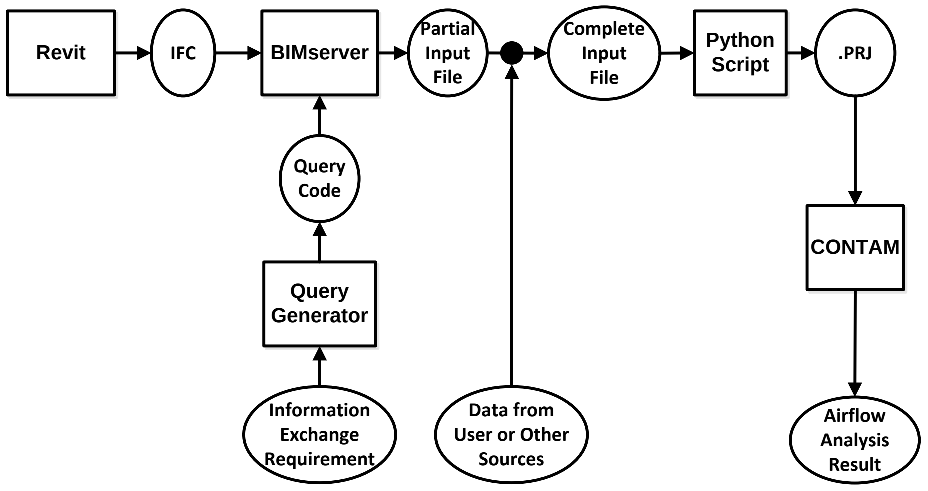

Figure 4-2. A Representative Workflow.

Figure 4-2 illustrates the testing workflow from Revit to the final airflow analysis result to test the functionality of Query Generator. BIMserver accepts the IFC file exported from Revit, and executes the automatically generated query code from Query Generator based on the information exchange requirement to query the partial input file. The partial input file for CONTAM is in JSON format. There is some missing information from the IFC model, so it requires the data from users or other sources. Combined with the missing information, the partial input file is changed into a relatively complete input file, also in JSON format. Then, a piece of Python script transforms the file in JSON format into .prj input file for CONTAM. Eventually, the airflow analysis result is generated by CONTAM.

Case Study 2: Accuracy of Extracted Geometry

Precision and correct modeling conventions are of the utmost importance for process. Therefore, this case is targeted to examine the accuracy of extracted and transformed geometry, which is implemented in the inner module: Transformer. The building design model contains 4 external walls, one of which is attached with a window and a door, 1 internal wall with a door, a roof, and a floor. As shown in Figure 4-3, all building elements are visualized in SketchUp. Compared with the original model, the location, direction are all correct. In terms of the relationship between spaces and surfaces, one space is shifted away from its original location, the related walls, floors and roofs are also moved together with the space in Figure 4-4. Accordingly, the accuracy of extracted geometry for the simple office building is checked.

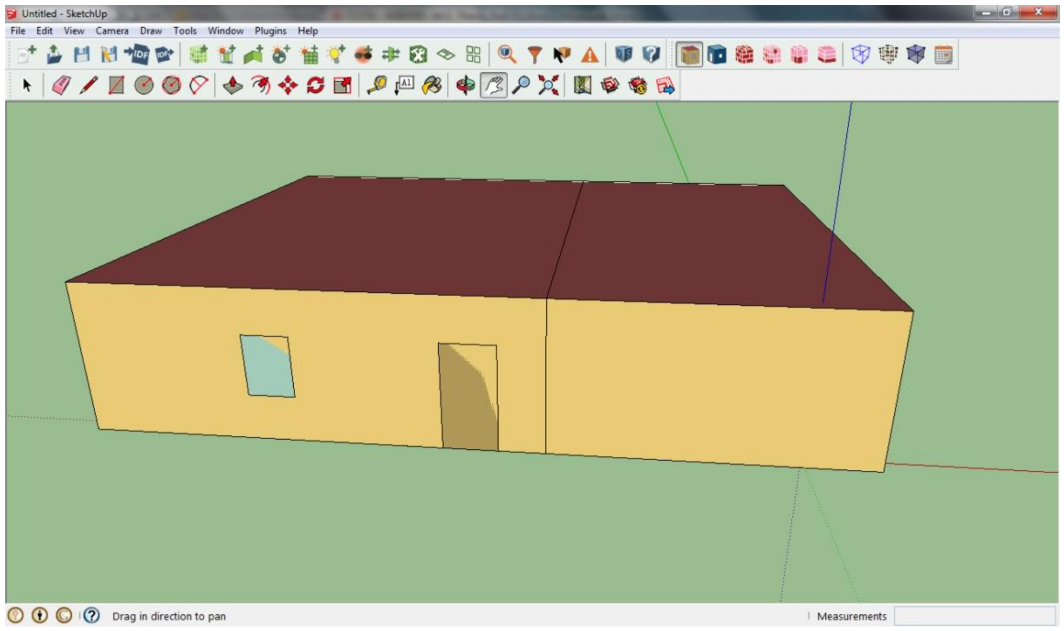

Figure 4-3. Simple Office Building—Original Model.

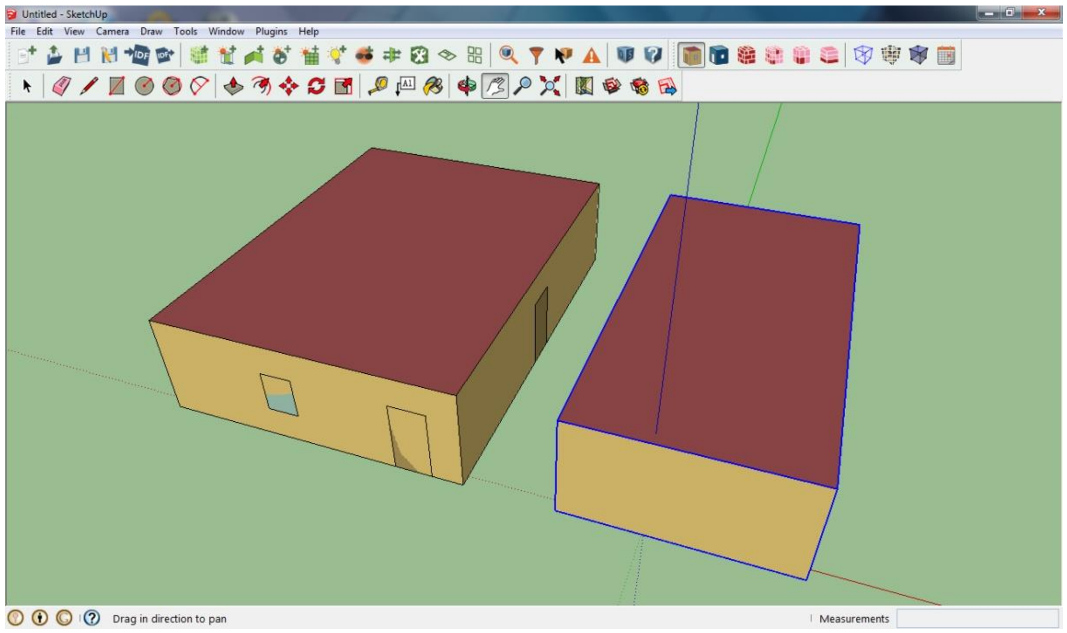

Figure 4-4. Simple Office Building—Separated Spaces.

Case Study 3: Basic Outline of Building 101 Identified by OpenStudio

Among all the building elements, IfcWalls, IfcWindows, IfcDoors, and IfcSlab (floor) have been successfully extracted transformed into OSM:Surfaces and OSM:SubSurfaces. It works well for simple office building as shown in case study 2. This case aims at testing the feasibility of complex buildings in real world. Figure 4-5 is a real picture of Building 101 in the Navy Yard in Philadelphia from the aerial perspective, and Figure 4-6 is the extracted OpenStudio model visualized in SketchUp of the same building. Constrained by technical transformation problem, roofs in complicated shape have not been transformed till now, so the uppermost level is missing in Figure 4-6. With additional effort, it is possible to further extract and transform the remaining elements.

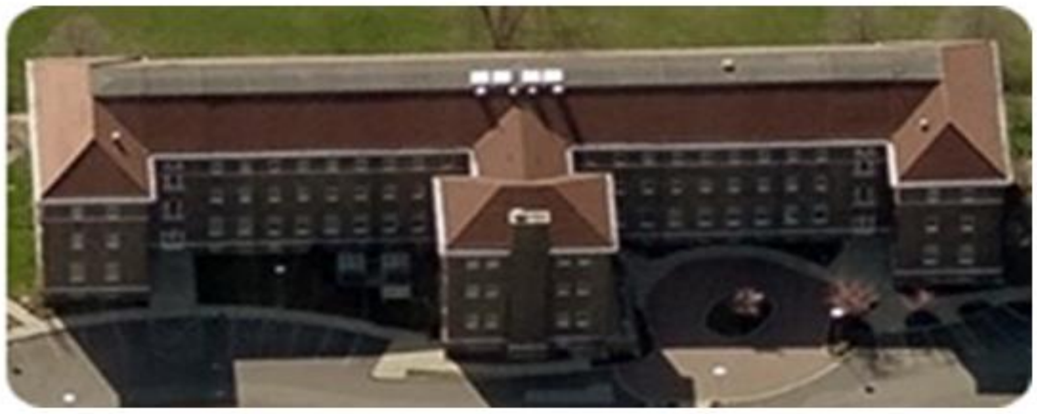

Figure 4-5. Building 101 in the Navy Yard in Philadelphia (EEB Hub, 2013). All rights reserved. Reproduced here for educational purposes only.

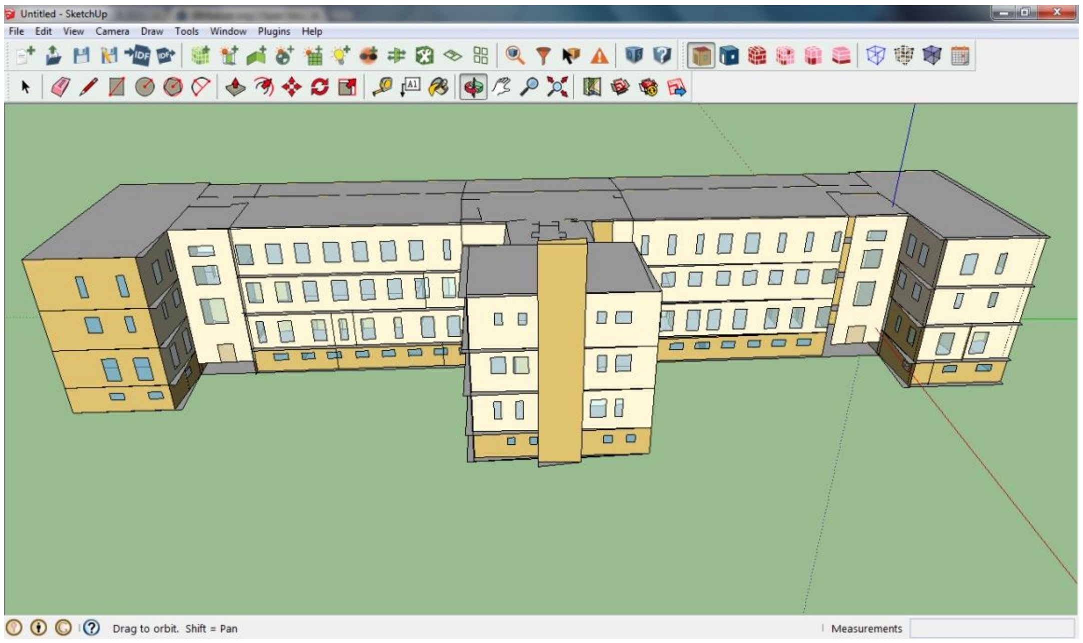
Untitled - SketchUp
File Edit View Camera Draw Tools Window Plugins Help
Drag to orbit. Shift = Pan
Measurements

Figure 4-6. Approximate Geometry Information Extracted from a Building 101 IFC Model.

Chapter 5

Limitations and Discussions

Although achievement in building a bridge between BIM design authoring tools and simulation tools has been achieved, the work is still a preliminary result which contains some limitations in real practice. First of all, the initial goal is to automate the whole process of data exchange from design to simulation, but to date, the process still involves some manual work, such as data import from or export to different tools. Secondly, IFC model contains all the necessary building information throughout the whole building lifecycle, but the data exported from Revit to BIMserver does not include the whole data set. It results in some missing information in OSM. Last but not least, the simplified data exchange model is based on the assumption that the input IFC file is a correct and valid model. To guarantee this assumption, an IFC model checker before BIMserver is necessary for the whole process.

Integrating Query Generator into BIMserver

Currently the Query Generator and BIMserver are running in their respective isolated workspaces. Users still need to copy and paste the automatically generated Java query code from the Query Generator into the GUI provided by BIMserver. It would be more efficient to eliminate human intervention completely by integrating the Query Generator into BIMserver as a plug-in.

BIMserver is an open-source project that allows open access, universal distribution, and subsequent improvement on top of it. The source code of BIMserver is organized by Eclipse Modeling Framework (EMF), which makes it possible to add customized modules within the framework to serve for the integrated data exchange model. The Query Generator is implemented in Java with the SWING GUI support package. At present, the Query Generator is designed independent from BIMserver deliberately because BIMserver keeps releasing new versions and the Query Generator is still under progress. The change of the interface of BIMserver would bring great impact to the Query Generator. Once the functionalities are all accomplished, the next step is to integrate the Query Generator into BIMserver as a plug-in, and automate the information extraction process.

Missing Information Required by OpenStudio

Simulation tools require the definition of input data. Simulation for a whole building simulation program such as EnergyPlus includes three distinct major parts: "the definition of building geometry and data related to it, the definition of Heating, Ventilating, and Air Conditioning (HVAC) equipment, systems and plant and data related to them, and definition of internal loads as well as use and operating schedules for the building" (Bazjanac, 2009, p. 1). The commercially available BIM design authoring applications do not include the HVAC system or operation schedules within the IFC export. Therefore, there is missing information extracted and transformed from IFC model to OSM. The missing information has a potential influence on the simulation analyses. For example, airflow analysis software requires HVAC systems to ensure thermal performance and design quality. Energy-related analysis with OpenStudio requires users to add construction layers to all surfaces, assign occupants/lighting/plug loads designed power and schedules to zones, add building location information and weather file, and assign basic HVAC systems and set points to the zones. All the information mentioned in the two cases are missing. Some of them, such as construction layers, and building location information can be extracted from the IFC model, but have not been extracted till now. The remaining part is not included in the original design model.

There are two solutions to solve this problem. Firstly, the Query Generator provides the users with a simple GUI to input the missing value for simulation analyses. The missing information should be able to dynamically adapt to different building design models. This solution does not apply to complex building models. For instance, if a building contains thousands of elements, users have to input the missing data for each element in the GUI. The second solution is to fill in the missing data offline. Instead of inputting data in the GUI, the application creates files for missing data for the building models. After completing the files, the application combines the user input with the previous partial input to generate a complete input file for simulation software.

These two approaches are comprised solutions. A simpler and better solution is to export the missing information that cannot be extracted from the current IFC model from the design authoring tools to the BIM data hub. This solution relies on third-party applications that support complete exporting from the design authoring tools.

Lack of Model Checkers

This thesis starts from the IFC model, and focuses on the transformation between the IFC model and the OpenStudio model. The information is extracted from the IFC model for simulation tools without any verification, and the transformed OpenStudio model is exported to OpenStudio directly. Once the simulation result is not reasonable, or any modules changes in the workflow, the whole process runs again. It is desirable to have some model checkers within the data exchange model to guarantee the appropriateness of both the IFC model and the OpenStudio model.

IFC Model Checker

The practice in this thesis assumes that the input IFC model exported from Revit to BIMserver is correct and valid. Base on that assumption, the required information is extracted and transformed into OpenStudio model. If the IFC model is not correct, the extraction from BIMserver and the transformation to OSM is wrong as well. In that case, adding an IFC model checker before importing the IFC model to BIMserver is a good way to guarantee the correctness and validity of the IFC model.

OpenStudio Model Checker

Currently, the OpenStudio model is imported into SketchUp for visualization view to check the transformed analytical geometric representation. If SketchUp fails to load the model with a list of warnings or errors, it is difficult to identify the corresponding elements with problems. Even if SketchUp succeeds in loading the model, errors can still occur during the simulation processes. Accordingly, an OpenStudio model checker can set up a set of rules to check the validity of the OpenStudio models. The following rules are listed as examples to support the checking of model correctness.

1. All spaces must be closed. Basically, a space includes attached walls, floors and roofs. The three building elements compose a space.
2. The space - surface relationship is stated by the space name in the OS:Surface; the surface - subsurface relationship is stated by the surface name in the OS:SubSurface.
3. A space cannot contain two identical surfaces. In other words, two surfaces having the same set of geometric vertices cannot exist in one space at the same time.
4. A surface or a subsurface must have at least 3 vertices.

5. All the vertices for this planar surface have to be in the same plane.
6. The subsurface cannot miss its base surface.
7. The subsurface have to have a face polygon.
8. For a shared wall, the model has to create two surfaces sharing the same location.
9. The construction layers have to be added to all surfaces.
10. Occupants/lighting/plug loads designed power and schedules have to be assigned to zones.
11. Building location information and weather file have to be added.
12. Basic HVAC systems and set points have to be assigned to the zones.

With the above rules, the OpenStudio model checker eliminates those errors before importing the generated model to OpenStudio.

Chapter 6

Conclusions and Future Work

This chapter summarizes the research questions in the AEC/FM Industry. It then presents the contribution of the work in this thesis, and closes with some discussions on the future work.

Summary

A high-performance building requires comprehensive whole-building simulation analyses to optimize energy consumption. The AEC/FM Industry experts often leverage information technology and various simulation tools to conduct analyses to assist with their decision-making in the whole building lifecycle. In the simulation data preparation process, the fragmented connection between BIM design authoring tools and simulation tools causes the difficulty for the domain experts to obtain the original building data in the design phase. Therefore, this process involves human intervention, which is inefficient and ineffective. BIM, as a bridge between the AEC/FM Industry and information technology, integrates the whole process throughout the building lifecycle. To build an integrated BIM data hub and combine more simulation tools to the system, this thesis presents a solution to connect BIMserver and OpenStudio together. In such architecture, OpenStudio manages the simulation tools and conducts various analyses; meanwhile, BIMserver offers the transactional query and data persistence service. Currently both OpenStudio and BIMserver have opened their programmable interfaces, which make it possible to integrate the two platforms. A data exchange model has been developed to build an information exchange bridge between BIMserver and OpenStudio, which enables different design and simulation tools that are connected to either of them to interoperate and exchange needed data. This thesis introduces the data exchange model in detail, along with the validation of the model. As preliminary work, the model has some limitations, which are also discussed in this thesis.

Contributions

The work from this thesis focuses on the data management services for the Energy Efficient Buildings (EEB) design, the goal of which is to develop and deploy to the building industry a state-of-the-art modeling platform that will integrate design, construction, commissioning, and operation.

In this thesis, the data exchange model builds a bridge between BIM design authoring tools and simulation tools. Although it is only preliminary work, the unified model shows the potential of supporting domain experts to easily conduct various energy-related analyses without reproducing the same building information. In the past, energy engineers spent most of their time on generating geometric representation from building design models for simulation data preparation. It takes only several minutes to extract the geometric representation using the data exchange model. Compared with manual generation in hundreds of hours, the automated extraction in several minutes increases the productivity and efficiency of energy engineers dramatically.

The data exchange model leverages two open-source platforms. It has been open-source released, which provides users with the opportunity to customize their requirement, and develop new functionalities on top of this model.

Future Work

The future work includes three main streams: (1) working out a complete geometric representation transformation model, (2) breaking through the limitations, and (3) further automating the process from BIM design authoring tools to simulation analyses.

The current progress only contains geometric information. Other information such as construction layers, HVAC systems, and operation schedules, is also required to generate a complete simulation model. Therefore, the extraction of the missing information, either from the original design models or from user input, is the first next step. As mentioned in Chapter, model checkers reduce errors before simulation running in OpenStudio. Accordingly, the IFC model checker and OpenStudio model checker are going to be implemented based on the set of rules listed in Chapter 5. After completing the missing information and the model checkers, the Query Generator can be integrated into BIMserver as a plug-in.

To further automate the simulation process, a new idea is brought up—Revit to OpenStudio in Two Clicks. The first click is a button in Revit to export the IFC model and start the process; the second click is a button in OpenStudio to import and create the OpenStudio model. According to this idea, another stream is to develop or find a Revit plug-in to upload IFC model to BIMserver, and to develop the BIMserver query for the OpenStudio 'Create Model' function. The objective is to establish an integrated eco-system around BIMserver with many tools and users.

References

Bazjanac, V. & Kiviniemi, A. (2007). Reduction, simplification, translation and interpretation in the exchange of model data. In *Proceedings of the 24th Conference. Bringing ITC knowledge to work*, 163–168.

Bazjanac, V. (2008). IFC BIM-based methodology for semiautomated building energy performance simulation. In *Proceedings of the 25th International Conference on Information Technology in Construction*. Santiago, CL, 292–299.

Bazjanac, V. (2009). Implementation of semi-automated energy performance simulation: building geometry. In *Dikbas, A., E. Ergen and H.Giritli (eds.), CIB W78, Proc. 26th conf., Managing IT in Construction*. Istanbul, TK 595–602.

Bazjanac, V., Maile, T., O'Donnell, J., Rose, C., Mrazovic, N. (2011). Data Enviroments and Processing in SemAutomated Simulation with EnergyPlus. In *CIB W078–W102: Proceedings of the 28th International Conference*. Sophia Antipolis, France.

Beetz, J., Berlo, L., Laat, R., and Helm, P. (2010). Bimserver.org—An Open Source IFC Model Server. In *Proceedings of the CIP W78 conference*. Cairo.

Borrmann, A., and Rank, E. (2009). Topological analysis of 3D building models using a spatial query language. *Advanced Engineering Informatics*, 23(4), 370–385.

buildingSMART, 2014a. Industry Foundation Classes (IFC) Homepage [WWW Document]. Retrieved March 12, 2014, from http://buildingsmart.org/.

buildingSMART, 2014b. Model View Definitions (MVD) Homepage [WWW Document]. Retrieved March 12, 2014, from http://www.buildingsmart.org/standards/mvd.

Domínguez, B., García, á. L., and Feito, F. R. (2011). Semantic and topological representation of building indoors: an overview. *CNKI Proc*. China.

Eastman, C., Teicholz, P., Sacks, R. and Liston, K. (2008). *BIM Handbook, Wiley & Sons*.

EEB Hub. (2013). Building 101 in the Navy Yard in Philadelphia. Retrieved December 31, 2014, from http:// www.eebhub.org/projects-list/navy-yard-building-101.

Ellis, P.G., Torcellini, P.A., Crawley, D.B. (2008). Energy Design Plug-in: An EnergyPlus Plug-in for SketchUp. In *Proceedings of the IBPSA–USA SimBuild Conference*. Berkeley, California.

Fleming, K., Long, N. & Swindler, A. (2012). The Building Component Library: an Online Repository to Facilitate Building Energy Model Creation. In *Proceedings of the 2012 ACEEE Summer Study on Energy Efficient Buildings*. Pacific Grove, Calif.

Foley, H.C. (2012). Challenges and opportunities in engineered retrofits of buildings for improved energy efficiency and habitability. *AIChE Journal*, March, 58(3), 658–667.

Guglielmetti, R., Macumber, D., Long, N. (2011). OpenStudio: an open source integrated analysis platform. In *Proceedings of the 12th Conference of International Building Performance Simulation Association*. Sydney, Australia.

Guglielmetti, R., Scheib, J. (2012). Challenges to Integrated Daylighting and Electric Lighting Simulation Methods in a Whole-building Energy Simulation Context. In *Proceedings of the 2012 Simbuild Conference*. August 2012, Madison, WI.

Heidarinejad, M., Wentz, J., Dahlhausen, M., Wang M., Yu, N., Lee, S., Benne, K., Macumber, D., Srebric, J., Wu, D., and Messner, J.I. (2014). A Hierarchy of Geometry Needs for Building Energy Models: Used for Bimserver and Openstudio Web Integration. Submitted to *the 2014 ASHRAE/IBPSA-USA Building Simulation Conference*. Atlanta, Georgia. September 12–14, 2014.

Hietanen, J. (2000). IFC R2.0 Object Diagram-Space to Wall connection. Retrieved March 12, 2014, from http://www.blis-project.org/private/objectdiagrams/IFCR2_SpaceToWallConnection_991026_jh.PDF.

Hitchcock, R.J., and Wong, J. (2011). Transforming IFC architectural view BIMS for energy simulation. In *Proceedings of the 12th International IBPSA conference*. Sydney, Australia.

International Alliance for Interoperability (IAI). (2006). IFC Model View Definition. Retrieved March 12, 2014, from http://www.blis-project.org/IAI-MVD/.

iwmnews (2009). A/E/C Industry Could Save $400B+ Annually with Building Information Modeling (BIM) Technology Says buildingSMART Alliance CEO. *Intelligent Workplace Management News*. Retrieved December 18, 2009, from http://www.iwmsnews.com/2009/12/aec-industry-could-save-400b-annually-with-building-information-modeling-bim-technology-says-buildingsmart-alliance-ceo/.

Jiang, Y., Ming, J., Wu, D., Yen, J., Mitra, P., Messner, J.I., and Leicht, R. (2012). BIM Server Requirements to Support the Energy Efficient Building Lifecycle. In *Proceedings of the 2012 ASCE International Conference on Computing in Civil Engineering*. Clearwater Beach, FL.

Jiang, Y., Ming, J., Wu, D., DeGraw, J., Lee, S., Jallow, A.K., Yen, J., Mitra, P., and Messner, J.I. (2014). BIM Enabled Energy Efficient Building Analysis: Improving Software Interoperability in the AEC Community. Manuscript in preparation.

Jones, B., Bogus, S.M. (2010). Decision Process for Energy Efficient Building Retrofits: The Owner's Perspective. *Journal of Green Building: Summer 2010*, 5(3), 131–146.

JSON.org. Introducing JSON. Retrieved March 12, 2014, from http://www.json.org/.

Kangaraj, G., Mahalingam, A. (2011). Designing energy efficient commercial buildings—A systems framework. *Energy Buildings*, 43, 2329–2343.

Khalili, A. and Chua, D. (2013). An IFC-Based Graph Data Model (GDM) for Topological Queries on Building Elements. *J. Comput. Civ. Eng*. 10.1061/(ASCE)CP, 1943–5487.0000331 (Jun. 5, 2013).

Klotz, L. (2011). Cognitive biases in energy decisions during planning, design and construction of commercial buildings in the United States: An analytical framework and research needs. *Energy Efficiency*. 4, 271–284.

Liu, F., Jallow, A.K., Anumba, C.J., Wu, D. (2013). Building Knowledge Modeling: Integrating Knowledge in BIM. In *Proceedings of the 30th International Conference on Applications of IT in the AEC Industry (CIB W78 2013)*. Beijing, China, October 9–12, 2013.

Liu, F., Jallow, A.K., Anumba, C.J., Wu, D. (2014). A Framework for Integrating Change Management with Building Information Modeling. In *Proceedings of the 15th*

International Conference on Computing in Civil and Building Engineering (ICCCBE 2014). Orlando, FL, June 23–25, 2014

O’Donnell, J., See, R., Rose, C., Maile, T., Bazjanac, V., Haves, P. (2011). SimModel: A domain data model for whole building energy simulation. In *IBPSA Building Simulation*. Sydney, Australia.

O'Donnell, J., Maile, T., Rose, C., Mrazovic, N., Morrissey, E., Reginier, C., Parrish, K., and Bazjanac, V. (2013). Transforming BIM to BEM: Generation of Building Geometry for the NASA Ames Sustainability Base BIM. *LBNL-6033E*.

Pollock, M., Roderick, Y., McEwan, D. & Wheatley, C. (2009). Building simulation as an assisting tool in designing an energy efficient building: A case study. In *Proceedings of the 11th international conference of building simulation Glasgow*, 1191–1198.

QueryGeneratorforBIMserver2.0. (2014). Retrieved April 1, 2014, from https://github.com/triangelfish/QueryGeneratorforBIMserver0.2.

Redmond, A and Smith, B. (2011). Exchanging Partial BIM Information through a Cloud-Based Service: testing the efficacy of a major innovation. In *Proceedings of the IBEA Conference*. South Bank University, London.

Revit. (2014). Retrieved March 12, 2014, from http://www.autodesk.com/products/autodesk-revit-family/overview.

Walton, G., Dols, W. (2010). CONTAM User Guide and Program Documentation. Retrieved December 25, 2010 from http://www.bfrl.nist.gov/IAQanalysis/docs/CWHelp30.pdf.

Weaver, E., Long, N., Fleming, K., Schott, M., Benne, K. and Hale, E. (2012). Rapid Application Development with OpenStudio. *2012 ACEEE Summer Study*. Pacific Grove, California.

Yu, N., Jiang, Y., Luo, L., Lee, S., Jallow, A.K., Wu, D., Messner, J.I., Leicht, R., Yen, J. (2013). Integrating BIMserver and OpenStudio for Energy Efficient Building. In *Proceedings of 2013 ASCE International Workshop on Computing in Civil Engineering (IWCCE)*. Los Angeles, CA.

Appendix

A. Code Snippets of OSM Generator 1

```
class Point{}
class OSMSurface{}
class OSMSubSurface{}
class OSMSpace{}

private void generateOutput(){
   for(OSMSpace osmSpace: allSpaces){
      outputContent.append("OS:Space,\n  ");
      outputContent.append(osmSpace.getSpaceName());
      outputContent.append(",              ! Name\n  ");
      outputContent.append(osmSpace.getTypeName());
      outputContent.append(",                     ! Space Type Name\n  ");
      outputContent.append(osmSpace.getDefaultConstructionSetName());
      outputContent.append(",                     ! Default Construction Set Name\n  ");
      outputContent.append(osmSpace.getDefaultScheduleSetName());
      outputContent.append(",                     ! Default Schedule Set Name\n  ");
      outputContent.append(osmSpace.getDirectionOfRelativeNorth());
      outputContent.append(",                     ! Direction of Relative North {deg}\n  ");
      outputContent.append(osmSpace.getxOrigin());
      outputContent.append(",                     ! X Origin {m}\n  ");
      outputContent.append(osmSpace.getyOrigin());
      outputContent.append(",                     ! Y Origin {m}\n  ");
      outputContent.append(osmSpace.getzOrigin());
      outputContent.append(",                     ! Z Origin {m}\n  ");
      outputContent.append(osmSpace.getBuildingStoreyName());
      outputContent.append(",                     ! Building Story Name\n  ");
      outputContent.append(osmSpace.getSpaceName());
      outputContent.append(" ThermalZone;  ! Thermal Zone Name\n\n");
   }

   for(OSMSurface osmSurface: surfaceList){
      outputContent.append("OS:Surface,\n  ");
      outputContent.append(osmSurface.getSurfaceName());
      outputContent.append(",                    ! Name\n  ");
      outputContent.append(osmSurface.getTypeName());
      outputContent.append(",                    ! Surface Type\n  ");
      outputContent.append(osmSurface.getConstructionName());
      outputContent.append(",                        ! Construction Name\n  ");
      outputContent.append(osmSurface.getSpaceName());
      outputContent.append(",            ! Space Name\n  ");
      outputContent.append(osmSurface.getOutsideBoundaryCondition());
      outputContent.append(",                ! Outside Boundary Condition\n  ");
      outputContent.append(osmSurface.getOutsideBoudaryConditionObject());
      outputContent.append(",                        ! Outside Boundary Condition Object\n  ");
      outputContent.append(osmSurface.getSunExposure());
      outputContent.append(",              ! Sun Exposure\n  ");
      outputContent.append(osmSurface.getWindExposure());
      outputContent.append(",             ! Wind Exposure\n  ");
      outputContent.append(osmSurface.getViewFactorToGround());
      outputContent.append(",                        ! View Factor to Ground\n  ");
      outputContent.append(osmSurface.getNumberOfVertices());

      int size = osmSurface.getPointList().size();
      if(size <= 0){
          outputContent.append(";                        ! Number of Vertices\n  ");
      }
      else{
          outputContent.append(",                        ! Number of Vertices\n  ");
          for(int i = 0; i < size; i ++){
              Point point = osmSurface.getPointList().get(i);
              outputContent.append(point.getX());
              outputContent.append(",");
              outputContent.append(point.getY());
              outputContent.append(",");
              outputContent.append(point.getZ());
              if(i < size - 1){
                  outputContent.append(",  ! X,Y,Z Vertex ");
              }
              else{
                  outputContent.append(";  ! X,Y,Z Vertex ");
              }
              outputContent.append(i+1);
              outputContent.append(" {m}\n  ");
          }
```

```
}
outputContent.append("\n");

List<OSMSubSurface> subSurfaceList = osmSurface.getSubSurfaceList();
for(OSMSubSurface osmSubSurface: subSurfaceList){
   outputContent.append("OS:SubSurface,\n  ");
   outputContent.append(osmSubSurface.getSubSurfaceName());
   outputContent.append(",                    ! Name\n  ");
   outputContent.append(osmSubSurface.getTypeName());
   outputContent.append(",                    ! Surface Type\n  ");
   outputContent.append(osmSubSurface.getConstructionName());
   outputContent.append(",                        ! Construction Name\n  ");
   outputContent.append(osmSubSurface.getSurfaceName());
   outputContent.append(",                 ! Surface Name\n  ");
   outputContent.append(osmSubSurface.getOutsideBoudaryConditionObject());
   outputContent.append(",                        ! Outside Boundary Condition Object\n  ");
   outputContent.append(osmSubSurface.getViewFactorToGround());
   outputContent.append(",                        ! View Factor to Ground\n  ");
   outputContent.append(osmSubSurface.getShadingControlName());
   outputContent.append(",                        ! Shading Control Name\n  ");
   outputContent.append(osmSubSurface.getFrameAndDividerName());
   outputContent.append(",                        ! Frame and Divider Name\n  ");
   outputContent.append(osmSubSurface.getMultiplier());
   outputContent.append(",                        ! Multiplier\n  ");
   outputContent.append(osmSubSurface.getNumberOfVertices());

   size = osmSubSurface.getPointList().size();
   if(size <= 0){
      outputContent.append(";                        ! Number of Vertices\n  ");
   }
   else{
      outputContent.append(",                        ! Number of Vertices\n  ");
      for(int i = 0; i < size; i ++){
         Point point = osmSubSurface.getPointList().get(i);
         outputContent.append(point.getX());
         outputContent.append(",");
         outputContent.append(point.getY());
         outputContent.append(",");
         outputContent.append(point.getZ());
         if(i < size - 1){
            outputContent.append(",  ! X,Y,Z Vertex ");
```

```
                }
                else{
                    outputContent.append(";  ! X,Y,Z Vertex ");
                }
                outputContent.append(i+1);
                outputContent.append(" {m}\n  ");
            }
        }
        outputContent.append("\n");
    }
  }
}
```

B. Code Snippets of OSM Generator 2

```
private void btnCreateModel_Click(object sender, EventArgs e)
{
    SaveFileDialog sfd = new SaveFileDialog();

    sfd.FileName = "in";
    sfd.DefaultExt = "osm";
    sfd.Filter = "OpenStudio Model (*.osm)|*.osm";
    sfd.CheckPathExists = true;
    sfd.OverwritePrompt = true;

    if (sfd.ShowDialog() == DialogResult.OK)
    {
        string fname = sfd.FileName;

        // Get user inputs from form
        double bldgLength = double.Parse(txtLength.Text);
        double bldgWidth = double.Parse(txtWidth.Text);
        double bldgHeight = double.Parse(txtHeight.Text);

        OpenStudio.Model model = new OpenStudio.Model();

        OpenStudio.Space space = new OpenStudio.Space(model);
        space.setName("MySpace");
```

```
// Create the Floor
OpenStudio.Point3dVector floorPoints = new OpenStudio.Point3dVector();
floorPoints.Add(new OpenStudio.Point3d(0, 0, 0));
floorPoints.Add(new OpenStudio.Point3d(0, bldgWidth, 0));
floorPoints.Add(new OpenStudio.Point3d(bldgLength, bldgWidth, 0));
floorPoints.Add(new OpenStudio.Point3d(bldgLength, 0, 0));

OpenStudio.Surface floor = new OpenStudio.Surface(floorPoints, model);
floor.setName("Floor");
floor.setSpace(space);
floor.setSurfaceType("Floor");
floor.setConstruction(construction);

// Create the front wall
OpenStudio.Point3dVector fwallPoints = new OpenStudio.Point3dVector();
fwallPoints.Add(new OpenStudio.Point3d(0, 0, 0));
fwallPoints.Add(new OpenStudio.Point3d(bldgLength, 0, 0));
fwallPoints.Add(new OpenStudio.Point3d(bldgLength, 0, bldgHeight));
fwallPoints.Add(new OpenStudio.Point3d(0, 0, bldgHeight));

OpenStudio.Surface fwall = new OpenStudio.Surface(fwallPoints, model);
fwall.setName("Front Wall");
fwall.setSpace(space);
fwall.setSurfaceType("Wall");
fwall.setConstruction(construction);

// Create the right wall
OpenStudio.Point3dVector rwallPoints = new OpenStudio.Point3dVector();
rwallPoints.Add(new OpenStudio.Point3d(bldgLength, 0, 0));
rwallPoints.Add(new OpenStudio.Point3d(bldgLength, bldgWidth, 0));
rwallPoints.Add(new OpenStudio.Point3d(bldgLength, bldgWidth, bldgHeight));
rwallPoints.Add(new OpenStudio.Point3d(bldgLength, 0, bldgHeight));

OpenStudio.Surface rwall = new OpenStudio.Surface(rwallPoints, model);
rwall.setName("Right Wall");
rwall.setSpace(space);
rwall.setSurfaceType("Wall");
rwall.setConstruction(construction);

// Create the back wall
```

```
OpenStudio.Point3dVector bwallPoints = new OpenStudio.Point3dVector();
bwallPoints.Add(new OpenStudio.Point3d(0, bldgWidth, 0));
bwallPoints.Add(new OpenStudio.Point3d(0, bldgWidth, bldgHeight));
bwallPoints.Add(new OpenStudio.Point3d(bldgLength, bldgWidth, bldgHeight));
bwallPoints.Add(new OpenStudio.Point3d(bldgLength, bldgWidth, 0));

OpenStudio.Surface bwall = new OpenStudio.Surface(bwallPoints, model);
bwall.setName("Back Wall");
bwall.setSpace(space);
bwall.setSurfaceType("Wall");
bwall.setConstruction(construction);

//Create the left wall
OpenStudio.Point3dVector lwallPoints = new OpenStudio.Point3dVector();
lwallPoints.Add(new OpenStudio.Point3d(0, 0, 0));
lwallPoints.Add(new OpenStudio.Point3d(0, 0, bldgHeight));
lwallPoints.Add(new OpenStudio.Point3d(0, bldgWidth, bldgHeight));
lwallPoints.Add(new OpenStudio.Point3d(0, bldgWidth, 0));

OpenStudio.Surface lwall = new OpenStudio.Surface(lwallPoints, model);
lwall.setName("Right Wall");
lwall.setSpace(space);
lwall.setSurfaceType("Wall");
lwall.setConstruction(construction);

// Create the roof
OpenStudio.Point3dVector roofPoints = new OpenStudio.Point3dVector();
roofPoints.Add(new OpenStudio.Point3d(0, 0, bldgHeight));
roofPoints.Add(new OpenStudio.Point3d(0, bldgWidth, bldgHeight));
roofPoints.Add(new OpenStudio.Point3d(bldgLength, bldgWidth, bldgHeight));
roofPoints.Add(new OpenStudio.Point3d(bldgLength, 0, bldgHeight));

OpenStudio.Surface roof = new OpenStudio.Surface(roofPoints, model);
roof.setName("Roof");
roof.setSpace(space);
roof.setSurfaceType("Roof");
roof.setConstruction(construction);

if (model.save(new OpenStudio.Path(fname), true))
{
   MessageBox.Show("Model saved to: " + fname);
```

```
            MainForm.ActiveForm.Close();
        }
        else
        {
            MessageBox.Show("Error saving model to: " + fname);
        }
    }
}
```