Technische Universiteit Eindhoven University of Technology

Construction Management and Engineering

A Linked Data approach for information integration between BIM and sensor data

2014-2016

Author:
Manlin Yu (s146559)

Graduation committee:
Prof.dr.ir. Bauke de Vries (TU/e)
Dr. dipl. ing. Jakob Beetz (TU/e)
Chi Zhang (TU/e)
Thomas Krijnen (TU/e)

Date of final presentation:
7th of July 2016

Table of Contents

Summary

Recent years, the trend to connect data from various sources becomes more and more obvious. The Semantic Web is developed to facilitate information linking, sharing and reusing across application, enterprise, and community boundaries. Linked Data, which lies at the heart of the Semantic Web, is able to make the Semantic Web a reality through creating interrelated data. In the building domain, Building Information Modelling (BIM) is popularly used to integrate building data among different life phases and different sub-disciplines. BIM is recognized as having a good effect in containing various building related information. However, the current integration between BIM and sensor data has some problems, like data interoperability troubles between data input and output. Since there already exist experiences in processing sensor data with the Semantic Web technologies, and the integrated result "Semantic Sensor Web" is argued for bring the usefulness of the sensor data to its full potential, this integration approach could also be implied in BIM and sensor data integration.

In this thesis, a Linked Data approach is developed to achieve information integration between BIM and sensor data for the facility managers' building performance analysing needs. The thesis is constructed of six chapters to clarify and verify this integration method. Chapter 1 discusses the research design background, the research questions and the research process. Chapter 2 is the glossary of this article. Chapter 3 holds the literature review of the current researches on BIM, sensor data, the Semantic Web and their integrations. Chapter 4 describes the methodology to integrate the BIM and sensor data by using Linked Data. Chapter 5 testifies this methodology through a case study from TU/e's Vertigo building. Last, chapter 6 gives the reflections of this integration method and provides some future research recommendations.

The methodology of this BIM and sensor data integration process is divided into six parts. The main flow path of the integration is collecting data, analysing data, transforming data, linking data, querying data and visualizing data. These technical processes are examined through the case study, and several query result visualizations are presented to help the facility managers analysing the building environment performances.

In the case study, there are four different data sources. Through the Linked Data approach, these four data sources with different data processing systems are integrated into one RDF model, and are convenient to be synthetically processed and queried without format or domain boundaries. It is testified that the Linked Data approach developed in this thesis, is effective to achieve the information integration between sensor data and BIM model, and could help the facility managers analysing the building operation performances through comprehensive information backup from different disciplines and data formats.

Abstract

The information integration between BIM and sensor data could provide strong building information back up for the facility managers when they perform the building operation monitoring. However, there are some limitations on the current integration researches, like focusing on specific software, or relying too much on IFC schema. This thesis raises a Linked Data approach to integrate BIM and sensor data. This approach is able to integrate the BIM model with sensor data from various sources regardless of their format or domain boundaries, and provide the facility managers an unified way to comprehensively analyse and visualize the building operation performances. A case study is performed at the end of the thesis to testify this BIM and sensor data integration approach.

1. Introduction

The central goal for the development of a building model in 2000s, was to include all the related building information and made it available throughout the whole building life cycle (Korpela et al. 2015). A forerunner of the building information modelling project in Finland named RATAS, defined its goal in 1988 as produced a model for structuring all the data on a specific building, for the use of design, construction and maintenance (Enkovaara et al. 1988). However, until now, it is recognized that this goal is not achieved, especially in the facility management domain. Annually, approximately $20 billion is lost in the US because of inadequate information access and interoperability problems in operation and maintenance phase (Newton 2004).

Within the operation and maintenance phase, aspects like asset management, space management, financial accounting and human resources management are all integrated together. In order to reduce inefficient building performance and operational cost, comprehensively identifying and managing all the data related to facility management process is necessary. This results in an increasing volume of data that is needed to be classified and stored. Moreover, the information required by different stakeholders in the facility management domain is different in categories and detailed levels. So, different subsets of data should be provided on the demands to the users with various organizational separation.

Building information modelling (BIM) is one of the most promising developments in the architecture, engineering and construction (AEC) industry, with the functions of decreasing project cost and delivery time, and increasing project productivity and quality (Azhar 2011). 3D visualization as well as the interrelated building element and attribute data are two important advantages of BIM (John et al. 2013). BIM provides an effective framework to capture the needed information and facilitate the interoperability between AEC domains. However, with the broader context of the AEC industry, nowadays, not all building related information is described in the BIM process. Other relevant information should also be integrated to optimize the building performance. Moreover, the Industry Foundation Classes (IFC), which is an open data exchange format for BIM, is not sufficient for the system interoperability outside AEC domains (Curry et al. 2013).

Sensor monitoring is an important dataset for building management during operation phase. With the sensor data, operation personnel is able to have an effective and efficient control over the building system. However, as a result of sensor data's time-related attribute, as well as various sensor data formats, there aren't many outcomes in integrating sensor data into BIM presentations. The current utilization of BIM focuses on its use as a static information repository and hardly concerns about building operationnal data and its monitoring. It could achieve greater value in making use of the building information from BIM and the time-related sensor monitoring information to provide a strong backup for further dynamic data analyses through building operation.

To represent the cross-domain information, the Semantic Web technology could be used on BIM and sensor information integration. And its core part--Linked Data, aims at building links between data from various data sets, so that data sets are not isolated data islands and

achieve data integration (Radulovic et al. 2015). The further developed concept “Semantic Sensor Web” (Sheth et al., 2008) from the Semantic Web, makes great contributions to the reusing and interoperability of sensor data. The combination of BIM and the Semantic technologies, along with the Semantic Sensor Web, make all the building data in BIM reusable outside its scope and easier to incorporate with dynamic building monitoring data to provide a holistic view of building operation.

In this research project, an intention of using Linked Data to combine BIM and sensor data, which enables monitoring and control of the building operation, is raised. And the research method is examined through a case study that generates from facility manager’s perspective to analyse the building operation results.

1.1 Problem definition

Facility managers make building controlling decisions with comprehensive building operation data. Among these operation information, data from BIM and sensors is a big part to support building performance analysis. Since sensor data files normally have large size and multiple data formats, nowadays they are separated operated with BIM. Moreover, storing dynamic and temporary sensor data in a static building model is easy to cause troubles on data operating.

However, integrating BIM model and sensor data is necessary, as sensor data is generated for building elements’ monitoring and needs to be pointed to its monitoring object, and also BIM model needs to present the operational performances through sensor data. The comprehensive building information attributes from BIM provide facility managers a holistic building information context to analyse monitoring data from sensors. The building information repository in BIM can also be used to store sensor-related information, like sensor manufacturers, sensor maintenance records and other supplementary sensor information, which can help facility managers grasping sensor using and strengthening building information repository. In addition, the 3D visualization of BIM gives facility managers an intuitive way to monitor building operation and judge the correlations between different building spaces’ performances. From the integration of the sensor monitoring data as well as related sensor supplementary information and the BIM model, a overall building information model is created for monitoring and maintenance purposes of facility managers.

To achieve the integration, sensor monitoring software or BIM applications are not sufficient. Sensor monitoring software does not have the building information repository function, while BIM and IFC format have inherent weaknesses in storing cross sources information. So, a third-party platform is needed to incorperate these two information technologies.

The Semantic Web, known as “an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation“ (Berners-Lee, 2001), is such an integrating platform. The Semantic Web provides a common framework to present multiple domains and formats information together with their inherent semantics, and makes connections between data to enable information sharing,

reusing and reasoning. The Linked Data approach, which is the core technology of the Semantic Web, is able to present sensor information in various original formats together with BIM model in one unified format. Moreover, with this approach, links can be built between building information and different sensor data to make the cross-domain information as a whole and to provide further query for facility managers. This provides comprehensive information backup for building operation and analysis.

In fact, the Sematic Web and Linked Data approaches can remedy many inherent structural disadvantages of BIM and the IFC format, and they can be well cooperated with sensor data. A more detailed discussion is presented in chapter 3 the literature review part.
To make a good combination, an integration can be developed to make full use of the three technologies: the Semantic Web, BIM and sensor data. In this research, a Linked Data approach is used to integrate BIM properties with sensor data, and help facility managers querying building performances and visualize them. A detailed description of the process will be presented in the following chapters.

1.2 Research questions

The information integration of the building properties and the sensor data monitoring is able to give facility managers a more concrete and holistic view of the building operation. Linking building operation statuses with the properties of the building elements, enables managers to make quick and balanced decision in different operation scenarios. In this research, BIM and IFC files are used to represent building objects. Linked data and the Resource Description Framework are employed to enable the data exchange between the sensor information and the building properties.

To achieve this data integration, during the integrating process, several research questions are addressed.

Main question:

- How to integrate a BIM model with sensor data to support facility managers in analysing building performances by using Linked Data ?

Sub questions:

- How to collect and convert different data sets into RDF format for building performance analyses?
- What kinds of links between BIM and sensor data can be created? How to achieve the linking?
- How can the integrated data help facility managers analysing building performances?

1.3 Research design

The research designis divided into six parts, as shown in figure 1.

The research begins from collecting data, both BIM model and sensor data, as well as other additional data that needs to be integrated. Part 2 analyses the data characteristics to have a basic outlook of the overall data structure and help performing the next parts. The following part 3, part 4 and part 5 form an iterative loop and may be repeated several times in the whole developing process, in order to test and modify each step's results. Part 3 focuses on turning all the original data sets into RDF format. Part 4 aims at creating links between data and integrating all the data and links into one graph. The query blocks developed in part 5 can be used to exam the results' accuracy of the previous two parts. Moreover, the developed query blocks can directly reflect facility managers' information demands. Part 5 will provide several options for the users to query the building's operational performance through the integrated information. The last part is to visualize the query results from the facility managers' query inputs. The query results will be visualized in a 3D representation of the building model with intuitive values indication.

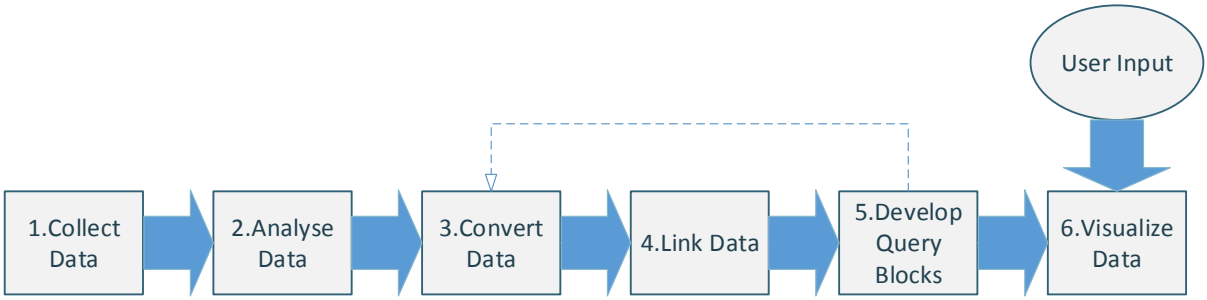


Figure 1 Research process

1.4 Expected results

This research develops a process to integrate BIM models and sensor data using a Linked Data approach. It also provides ways for facility managers to analyse the building operation performances through performance visualization. Through a case study, the whole process and methodology will be verified.

2. Glossary

AEC	The abbreviation for Architecture, Engineering and Construction.
Building Information Modelling (BIM)	A process to create and manage digital representations of buildings.
EXPRESS	A standard data modelling language for product data.
IfcOpenShell	An open source software library that helps users working with the IFC file format.
IfcOWL	An ontology for building data from IFC.
Industry Foundation Classes (IFC)	An exchange file format for BIM data.
Linked Data	A method to publish semantic data.
Ontology	A formal naming and definition of the types, properties, and interrelationships of the entities in a particular domain.
PMV	The abbreviation of the Predicted Mean Vote on thermal comfort calculation.
PPD	The abbreviation of Predicted Percentage of Dissatisfied on thermal comfort calculation.
Resource Description Framework (RDF)	A general-purpose language for representing information in the Web.
Resource Description Framework Schema (RDFs)	A data-modelling vocabulary for RDF data.
Smart Appliances REFerence (SAREF) ontology	An ontology to present semantic data for smart appliances in buildings and households.
Semantic Sensor Web	A marriage of sensor and the Semantic Web technologies.
SPARQL Protocol and RDF Query Language (SPARQL)	An RDF query language for databases.
The Semantic Web	A framework allowing data to be shared and reused across application, enterprise, and community boundaries.

The World Wide Web Consortium (W3C)	The main international standards organization for the World Wide Web.
Turtle	A format for expressing data in the RDF data model.
Uniform Resource Identifier (URI)	A string of characters used to identify a resource.
Uniform Resource Locator (URL)	A reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.
Web Ontology Language (OWL)	A Semantic Web language designed to represent knowledge about things and their relations.
WGS84 Geo Positioning Ontology	A vocabulary for representing latitude, longitude and altitude information in the WGS84 geodetic reference datum.

3. Literature Review

In recent years, sensor monitoring is increasingly adopted in various industries. Great benefits have been brought in disciplines like Geographical Information Systems (GIS), weather forecasting, traffic planning and management, and smart homes. Countless sensors with different types and capabilities are distributed across the globe. The weak integration and communication between these sensors lead to a dilemma of "too much data, but not enough knowledge" (Sheth et al., 2008). Gartner predicted that "by 2015, wirelessly networked sensors in everything we own will form a new web. But it will only be of value if the 'terabyte torrent' of data it generates can be collected, analysed and interpreted" (Raskino et al., 2005). Regardless of how much of Gartner's prediction comes true, plenty of researches appeared in increasing the interoperability and linkage of sensor data.

In this section, current studies in the fields of sensors, the Semantic Web and BIM, as well as their pairwise integrations are briefly discussed. A theoretical certification of combination between Semantic Sensor Web and BIM for helping building management are conducted in the end.

3.1 Limitations in Building Information Modelling (BIM) and IFC

Building Information Modelling is one of the most powerful developments in the AEC industries. With BIM technology, various virtual models from 2D drawings to 3D presentations can be accurately constructed. It can also be counted as a central repository of the building data served for all the project stakeholders through the whole project lifecycle (Eastman et al., 2011). The Industry Foundation Classes (IFC) is a commonly used collaboration format for BIM. It is an EXPRESS schema within the STEP family of standards (ISO 10303) (Liebich et al., 2006) that is developed by BuildingSMART, formerly known as the International Alliance for Interoperability (IAI). It is developed for defining extensible sets of consistent data to represent building information for exchanges between AEC software applications (Eastman et al., 2011).

However, within the wider context of the building industry, BIM is only one silo of information and other relevant information must also be integrated with it (Curry et al., 2013). Moreover, during implementing IFC models into practical applications, several serious problems appeared.

Pauwels et al. argued that IFC could only describe information within its own schema to enable the interoperability among other IFC files. When it comes to remote domains such as geographic information, or niche domains as well as research faces within building industry, its limitation in expression ranges appears. Also, according to various BIM based designs and implementations, multiple descriptions of the same information may exist in different IFC files. This requires extra design efforts from software engineers to reuse the information (Pauwels et al., 2011).

Fischer & Kam mentioned that the information partitioning during the IFC model transformation was not available, so that stakeholders always had to receive the whole size model instead of only picking up the related information. Also, versioning and controlling user rights through the file exchange is practically impossible (Fischer & Kam, 2002).

Beetz et al. addressed that the IFC format was not based on a mathematically rigid theory like OWL and lack of formal rigidness. The EXPRESS modelling language used by IFC has limitations in resources reuse and interoperability. In addition, within the STEP world, some structural shortages like file-based indexing and attribute scoping local to entity definitions cause obstacles for IFC built-in distribution (Beetz et al., 2009).

To extend the development possibilities of IFC, the Semantic Web technologies appears as a strong support to facilitate building information interoperability, interaction and rigidness.

3.2 The Semantic Web

Besides the creation of knowledge through observation, networking of knowledge is the foundation to generate new knowledge. Integrating new knowledge into the existing information space of the web, and exploiting semantics to create an overall knowledge network to bridge the islands between people, organizations and systems, is a new way to promote innovation and increase productivity (Decker & Hauswirth, 2008).

The Semantic Web, as defined by the W3C Semantic Web Activity, is a common framework based on Resource Description Framework (RDF), to enable data sharing and reusing across applications, enterprises, and community boundaries[1]. The Semantic Web is about linking, and the feature is that the links themselves in the Semantic Web all have specific meanings. As the RDF language defines, all the information described in the Semantic Web is presented in the basic unit “triple”: subject, predicate, object. These three parts in triples have their own meanings. This equips every information part in the Semantic Web with semantic meaning and is convenient for machine reasoning.

There are two ways to build the Semantic Web, 1) linking the information that exists within documents, 2) and allowing data itself to be on the Web[2]. There are two main steps to use the Semantic Web, 1) providing formal and machine-readable specifications for the conceptualized information communities, i.e., by creating ontologies, and 2) using inference engines to explore implicit relations, facts and potential contradictions (Janowicz et al., 2010).

Linked Data, which lies at the heart of the Semantic Web, refers to the collection of interrelated data sets with standard and manageable formats on the Web[3]. In summary, Linked Data is about using the Web to create semantic links between data from differentsources. These sources may be as diverse as databases maintained by two

[1] https://www.w3.org/2001/sw/

[2] https://journals.ala.org/ltr/article/view/4669/5539

[3] http://www.w3.org/standards/semanticweb/data

organisations in various geographic locations, or just heterogeneous systems within one project (Bizer et al., 2009). The Linked Data concept was first introduced by Tim Berners-Lee in 2006 (Berners-Lee, 2006). And he suggested four main principles to publish Linked Data:

- Using URIs as names for things
- Using HTTP URIs for better information processing
- Using standards like RDF and SPARQL to provide useful information
- Including links to other URIs for discovering

Basic Semantic Web technologies include data modelling language RDF (Web, 2011), the ontology representation model RDF Schema (Brickley & Guha, 2000), the added logical formalism Web Ontology Language (Patel et al., 2004), as well as RDF query language SPARQL (Prud & Seabome, 2008). The cooperation of these technologies helps integrating of and reasoning on data on the web.

Through relating data attributes and metadata features to other resources on the web of data, the Semantic Web users will be able to integrate physical world data and logical world data to do things like drawing conclusions, creating business intelligence, enabling smart environments, supporting automated decision making systems, etc. As an important component of “Web 3.0” (Shannon, 2006), the Semantic Web has a potential and bright future.

3.3 BIM and the Semantic Web

BIM is helpful for 3D visualisation, clash detection but less considered in information communication between applications for usages like building analyses (Becerik & Rice, 2010). The IFC format is just developed for the communication of building information among different BIM applications. Nevertheless, as a result of the broad correlations and various domains of building industry, applications for building design and management are not only limited to the IFC exchange format. Information distortion and/or loss exists not only in IFC conversion from or to other file formats, but also within the IFC-based building information re-use(Pazlar & Turk, 2008). It is impossible to reuse the IFC data in an application that employs even a slightly different schema, as well as describe information exceed pre-defined schema (Pauwels et al., 2010).

The root of these IFC shortages lies in the nature of EXPRESS modelling language in converting semantically rich information. And the Semantic Web technology can bridge this through presenting multiple domains and formats information together with their inherent semantics (Bemers et al., 2001). With the Semantic Web technology, it is applicable to connect multiple building information schema together with the IFC schema, instead of focusing on one central IFC standard. The web of information becomes the central information source to freely provide information and services according to every stakeholder’s need.

Many researches have shown efforts in incorporating IFC with the Semantic Web technology. For example, Beetz et al. developed an approach to achieve the translation from IFC schema to Semantic Web graph (Beetz et al., 2009). Yurchyshyna showed how semantic queries could

conduct building conformance checking against building codes on BIM models (Yurchyshyna & Zarli). Pauwels developed a semantic rule checking method for BIM model on acoustic performance (Pauwels et al., 2011).

Underwood and Isikdag raised three attributes that affected the semantic query's success rate when integrated the Semantic Web and BIM:

- The integration level of distributed building information,
- The success level in deriving information mass from multiple loosely coupled Web resources,
- How well the query can be interpreted, as well as reasoning and retrieval can be accomplished, upon the interpreted query (Underwood & Isikdag, 2011).

3.4 Sensor Semantic Web

Recent years, considerabl3 research efforts have been put into large scale sensor networked projects, like SensorWeb[4] and the SENSEI project[5]. Meanwhile, several industry specifications related to sensors, sensor data models, and sensor web services have been developed by the Open Geospatial Consortium (OGC) (Sheth et al., 2008). The Sensor Web Enablement (SWE) initiated by OGC is responsible for standards development that makes sensors and their gathered data available on the Web (Janowicz et al., 2010). Some of the language specifications are listed below:

- Observations and Measurements (O&M). Standard models and XML schema that encode observations and measurements from a sensor. Sensor data can be archived or real-time.
- Sensor Model Language (SensorML). Standard models and schema that describe sensor systems and processes. For example, provide information for locating sensor observations and processing low-level sensor observations.
- Transducer Model Language (TML). Standard models and schema that describe transducers and support real-time sensor data transformation (Sheth et al., 2008).

However, most of the sensor data descriptions that have been developed are based on the XML language. This leads to a significant weakness in developing sematic interoperability and making links between described resources and existed knowledge (Wei & Bamaghi, 2009).

To overcome these limitations of XML-based sensor description, semantic web technologies have been applied. Semantic annotation using sensor-domain ontologies that based on Linked Data principles has been applied to represent sensor web information. And the knowledge discovering from the annotated sensor data has two levels: the low-level is for sensor observation, discovery and retrieval, and the high-level is for service planning and recommendation. The two-level annotated sensor data exploring can be performed through Semantic Web technologies like RDF, RDF Schema, Web Ontology Language and SPARQL Protocol and RDF Query Language to support interoperability. Semantic annotation can easily

[4] http://research.microsoft.com/en-us/projects/senseweb/
[5] http://www.ict-sensei.org/

integrate abundant sensor information, and logical reasoning using OWL and SPARQL can conduct advanced sensor data retrieval and query tasks (Wei & Bamaghi, 2009).

As a result, the concept of "Semantic Sensor Web" has been brought forward by Sheth et al. In his theory, sensor data is annotated with semantic metadata to increase interoperability and to provide contextual information for situation representation. A semantically rich sensor network will provide three types of semantic metadata for analysing: spatial, temporal and thematic. Spatial metadata is the information about sensor locations and geographical reference system, local system or named locations. Temporal metadata is the information about time instant or interval of sensor data capturing. Thematic metadata is about sensor observation's real-world states, like objects and events (Sheth et al., 2008).

Besides the ontology developed by Sheth et al as mentioned above, many other sensor data ontologies has been developed. OntoSensor developed a deep sensor ontology that extended the IEEE Suggested Upper Merged Ontology (SUMO)[6] and enables sensor parts compatibility determination, dynamic sensor selection and tasking (Russomanno et al., 2005). Eid et al. proposed a two-layer prototype ontology that also excerpted SUMO. This ontology uses SUMO as a root definition and adds two sub-ontologies: the sensor data sub-ontology and the sensor hierarchy sub-ontology (Eid et al., 2007). Kim et al. raised a service-oriented sensor ontology that is developed based on ontologies like SUMO and OntoSensor. It enables service-oriented services for future ubiquitous computing (Kim et al., 2008). Bamaghi et al. proposed a sensor data ontology that is built on Sensor Web Enablement (SWE) and SensorML data model, to uniformly described the semantic relationships and operational constraints (Bamaghi et al., 2009). Every ontology has its own focused situation and also has some limitations such as that they can not present sensor data specification details or complexed relationships between sensor data. Based on different sensor data use cases, different ontologies should be chosen. More comprehensive and micromesh ontologies will be further proposed.

As Wei and Barnaghi concluded: "Semantic Web technologies is an ideal choice to represent, manage, store, analyse, and reason over the observed sensor data, enhancing interoperability among heterogeneous networks to build Semantic Sensor Web, and to bring usefulness of the sensor data to its full potential" (Wei et al., 2009). The Semantic Sensor Network will be the future trends for sensor monitoring.

3.5 BIM and sensor data

BIM models are favourable for their interactive 3D views of building plans, sections and elevations (Azhar et al., 2008). Visualizing the sensor implementation in BIM applications enables system users to see the exact geographic information of sensors, which provides a more intuitive sense for building operation. What's more, BIM is identified as "future IT solution" (HM Government, 2012). The "object-oriented" basement (Yan & Damian, 2008) and

[6] http://www.ontologyportal.org/

whole life-cycle information management of BIM enable a more holistic building information context for facility managers to understand sensors' feedback and conduct decision making.

The solution for BIM and sensor data integration for facility management has been researched and practiced for several years (Riaz et al., 2014). According to Riaz's literature review, from 2005, there have been continuous attempts in combining BIM and sensor technologies for sensor-monitored environments to help building management. For example, Yin proposed an improvement to building services controlling through the Building Management System (BMS) and BIM (Yin, 2010). O'Flynn et al. designed a miniaturized Wireless Sensor Network mote for building monitoring, and used it as an input tool for BIM models (O'Flynn et al., 2010). Woo et al. developed a prototype of BIM-based Baseline Building Model (B3M) along with Sensor Network to monitor building environments (Woo et al., 2011). Cahill et al. monitored the sensor data and identified it in IFC file (Cahill et al., 2012). Riaz et al. developed a prototype system using BIM to present wireless sensor data in order to help building monitoring (Riaz et al., 2014).

However, some of these BIM and sensor integrated researches like Yin's and O'Flynn's are theoretical developments and only establish a basic framework, but lack of practice and detailed research. Some researches like Woo's and Riaz's are focused on one BIM software platform and may had problems when convert the integrated data into other BIM applications. Researches like Cahill's are focused on relying IFC schema so that the result value is partly blocked by the IFC schema's inherent weakness, like having limited interoperability between different applications as well as with other data resources. So, purely combining sensor data with BIM is not sufficient, additional information integration and representation technologies have to be added.

3.6 Using Linked Data to integrate BIM and Semantic Sensor Web

As analysed above, the IFC format is able to contain whole life-cycle building information, but has limitations in resources expansion, reuse and interoperability. The Semantic Web is good at integrating various information resources as well as generating and sharing new knowledge. If BIM is combined with the Semantic Sensor Web, web resources like real time sensor data, geographical reference information and weather information can be linked with building models. When the information provided by the Semantic Sensor Web integrates with the Building Information Modelling, the full building information context can be generated and make the decision making results more accurate and up-to-date. Moreover, as all the building and sensor related information is put on the web in terms of triples, data sharing, reusing and interoperability between different applications and different users will have lower barriers. Last but not least, the integrated real-time sensor network information with BIM can help to provide synthetical queries based on overall building and environment context. Semantic queries such as "Would you provide me indoor quality for the in-use meeting rooms on the fifth floor? ", or "Would you provide me the thermal comfort index distribution for the whole building? ", or "Would you provide me the number of working elevators in the building

between 12:00 till 14:00? ”, can provide strong back up for quickly decision makings on building management for facility managers.

The published researches on the integration of BIM and Semantic Sensor Web are not rich. Underwood and Isikdag mentioned Semantic Sensor Web with BIM as emerging technology for BIM 2.0 (Underwood & Isikdag, 2011). Corry et al. utilised Linked Data between sensor network, BIM and other data resources to perform thermal comfort assessment of a building space under Performance Framework Platform (PFP)(Corry et al., 2013). But Corry’s research is focused on the PFP OWL ontology development and the reconfiguration of an existing PFP tool using this ontology. Corry later further developed the PFP based on BIM, sensor network and other resources information, and raised a Performance Management Framework for a holistic building performance management(Corry, 2014).

Future research directions can be achieving more practical use case studies in integrating Semantic Sensor Web with BIM and applying the integrated results to more applications, for example, visualization platforms or building management systems, in order to examine the benefits that Linked Data can bring to sensor network and BIM, as well as explore further improvements for this integration method.

4. Methodology

In chapter 1 the introduction part, the research questions and research design of this thesis are introduced. This methodology chapter is focused on describing the technical developing process for the BIM and sensor data integration by using Linked Data.

The methodology model is presented in figure 2. The whole technical process includes six major parts, as section 1.3 shows. Every part in the process also includes several steps to complete. Each of these steps will be introduced in the following sections.

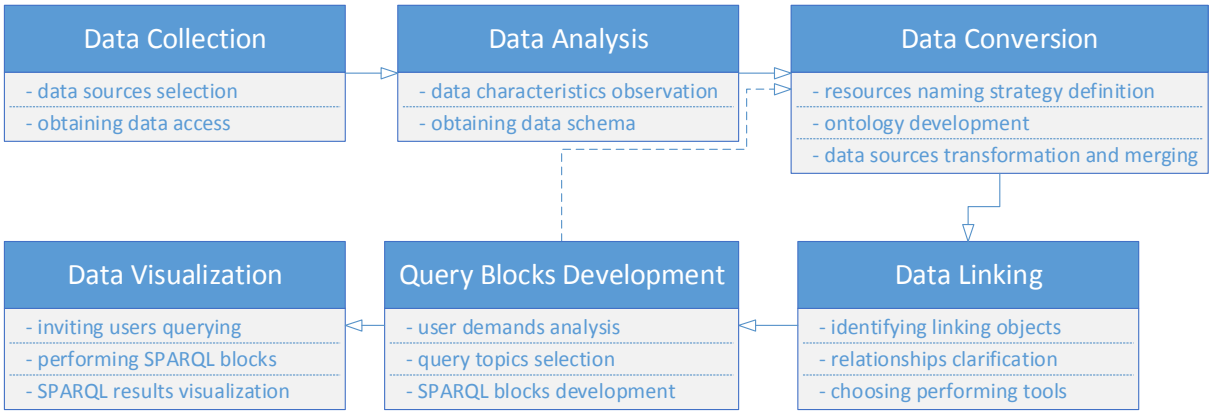


Figure 2 Methodology

4.1 Data collection

Data collecting is the first step of Linked Data generation and can easily affect the whole process results. The aims of data collection is selecting suitable data sources to perform the integration task.

4.1.1 *Data sources selection*

The key criterion for data selection is that the data can meet the facility managers' specific information needs, like interior environment monitoring, facility operation status reflection. Apart from the facility managers' needs, the accuracy, continuity and stability of the data should be an important criterion. The synthesis of these two criteria can help selecting useful and suitable data sources for the information integration. To well serve the facility managers' information needs, not only direct data that meets the needs, but also auxiliary and exegetic data should be collected.

4.1.2 *Obtaining data access*

Obtaining data access has two layers of meaning. The first layer of meaning is to find correct channels to receive the data. Most of the public domain data resources can be accessed without obstacles, but some private or organizational owned data may have identity requirements. Also, the platforms that the information is provided from may be different: file, web site, database, etc. Finding the suitable methods according to different platforms to save the data is the key point.

Obtaining data access also means collecting data with legal allowance. Authoritative data publishers may provide different applicable licenses for different end users, different using purposes and different processing methods. To avoid legal conflicts, it is necessary to identify the authoritative data publishers and related data using license.

4.2 Data analysis

When all the data resources have available legal licenses for further processing, the next step is to look into the data sets themselves.

4.2.1 *Data characteristics observation*

Data analyses begins from analysing how these datasets are structured and organized. The obtained formats of datasets may be heterogenous, and they have to be analysed in order to have a basic understand of what information is provided and it is provided in what kind of form. To help the following data processing, some of the data formats may need to be transformed. The preferred data formats can be determined according to the data organizing structure and the formats of other data sources.

4.2.2 *Obtaining data schema*

Obtaining data schema aims at having a basic understand of the various data-sets' major concepts, as well as information overlap and expression differences. Analyse the relations among different major concepts to have a combined relational graph. This can help performing the following conversion and linking tasks, and also justify again that the joint data is enough to meet the facility manager's information needs.

4.3 Data conversion

The converting data part is mainly about data transformation to RDF format. With the purpose of unifying data format and preparing for data linking, the data conversion is structured into three steps.

4.3.1 *Resource naming strategy definition*

In the four main principles of publishing Linked Data (Berners-Lee, 2006), Uniform Resource Identifier (URI) is required for naming things. For the URI naming, there are several forms and guidelines. For example, there are two basic forms of URI— hash URI and slash URI (Radulovic et al., 2015). There are also different guidelines that are helpful for URI design, like 10 rules for persistent URIs (SEMIC, 2012), and Providing and Discovering URI Documentation (Ree et al., 2012). However, for choosing a basic form of URI, the advantages and disadvantages of each basic form and also the preferred URI form for the data transform tools that will be used afterwards should all be considered. Besides the URI forms, clearly and succinctly showing the relationships and hierarchy between various resources is essential for URI design as well.

4.3.2 *Ontology development*

An ontology is a formal specification of a shared conceptualization (Gruber, 1993). In computer science and information science, it is used to name and define types, properties, and interrelationships of entities from a particular domain. Radulovic raised seven steps to develop a well-designed ontology (Radulovic et al., 2015). The main idea of his ontology development method is referenced in this ontology developing step.

Developing an ontology to describe resources and relationships from a merged domain, starts from considering the overall data content and structure, as section 4.3.1 describes, in order to grasp the main concepts forming the ontology and their connections. After that, a search for existing ontologies and the selection of some of these for reuse should be conducted. Reusing existing ontologies can help building a new ontology structure and makes the new ontology more accessible. For the information not defined in existing ontologies, new classes or properties should be created to represent it.

4.3.3 *Data sources transformation and merging*

The resources naming and ontology developing are all preparing for the data transformation step. As Radulovic et al. suggested, select a RDF serialization(RDF/XML, Turtle, N-Triples etc.) as data transformation format. Then select the transformation software to achieve the RDF conversion based on the input and output data formats that the software supports. Since data sources may be collected with different formats, like spreadsheet, XML, and IFC, several transformation tools may be needed. Afterwards, perform the data transform to RDF format, and merge the transformed RDF files into one model. This can be produced through Java RDF API or other software like Google Refine. The evaluation of the data transforming and merging result can be executed by SPARQL queries in the following part. If the SPARQL queries go smoothly and get correct results, the data transforming and merging part works well.

4.4 Data linking

Data linking aims at creating links between the RDF data that comes from different data sources. The data resources' content analysis and schema extraction have been performed in the data conversion part. Full preparation has been made for the data linking.

4.4.1 Identifying linking objects

The RDF data from different data sources are isolated in the merged model. Selecting which objects from various sources to perform linking is dependent on the goal of the performance task that developed for facility managers. The new created links should help users to navigate between different information islands and to collect needed information. Taking into account the data ontology, choose the linking objects that can help building most direct and convenient relationships between RDF data from different sources.

4.4.2 Relationships clarification

After identifying linking objects, clarifying resources relationships through suitable properties is the next step. The property selection can take a reference of existing former ontology vocabularies. If no satisfied properties were defined, then create some clear and easy-understanding properties by yourself.

4.4.3 Choosing performing tools

Use a tool to perform the linking creation among RDF data. There are many tools providing RDF linking function, like LN2R, Google Refine, Jena Apache API, etc. Different tools have different data requirements and function configurations, like RDF serialization limitation, data structure requirements and data size requirements. Selecting a suitable tool based on the data attibutes can make the linking process easier.

4.5 Query blocks development

The query design reflects facility managers' information demands that are going to be met in this research on building operation controlling. It is designed based on available data and user demands that help managers to grasp building performances. Every query performs a specific topic. All the composed coding components for this query topic are collected as a whole and is called a "query block", as a packaged block is more convenient than distributed codes for facility managers to call. The query blocks not only conduct SPARQL queries, but can also use query results to do some graphic comparison and calculations.

SPARQL queries can be seen as a good evaluation of the results from data converting and data linking parts. They are able to test whether the ontology design is feasible, whether the RDF data transformation result is able to show original data structure, whether the data merging and resources linking work well, etc. Once the query results reflect some problems existing in the previous parts, the process begins again from the data conversion part needed to be examined again.

4.5.1 User demands analysis

The data sources in this research are sensor data and a BIM model. The users is focused on facility managers and aims at helping them mastering building operation performances. So, analysing facility managers' possible information demand based on available sensor data and building information is the basement to design query blocks. The analyses can base on data content, user's interview or literature study.

4.5.2 Query topics selection

After user demand analysis, pick up several topics that are daily necessary for facility managers' building operation monitoring or synthetically conclusive analyses helping facility managers on building performance assess.

4.5.3 SPARQL blocks development

When the query topics are fixed, the next step is developing SPARQL blocks. The blocks include the SPARQL query part and the query results processing part. The SPARQL query part extracts information among direct or indirect resources relationships. And the query results processing part performs further data deductions to help facility managers understanding building operation statuses. The different blocks are based on different query topics and all packaged for users' callings from next visualization step.

4.6 Data visualization

Directly reporting to a facility manager with numeric query feedback lacks intuitive and necessary building information context. Visualizing query results within the building model or diagram presentation helps users easier to make quick and reasonable management decisions.

4.6.1 Inviting users querying

To make the query process user-friendly and to free facility managers from having to retrieve the data they wanted, the visualizing step first invites users to input a query topic. Corresponding input restrictions will come along with the invitation to help users standardizing query inputs.

4.6.2 Performing SPARQL blocks

As mentioned in the query blocks developing part, the various query blocks are capsulized in order to quickly response facility managers' different query requests. Once the facility manager inputs a query request, the query operating platform will execute the corresponding SPARQL block and generate the analysed results.

4.6.3 SPARQL results visualization

The visualization of daily building performance monitoring is developed based on IfcOpenShell[7], an open source software library that helps developers working with IFC file format. IfcOpenshell is used to make numeric query results reflected in various coloured building spaces based on IFC file. The different colours for the building spaces represent different numeric result ranges.

The visualization of conclusively building performance assess is present in histogram, which helps facility managers comparing the building performance behaviours.

The detailed visualization design will be elaborated in the case study.

[7] http://ifcopenshell.org/

5. Case Study

In this chapter, a real case is used to apply the methodology described in chapter 4 and aims at achieving sensor data and a BIM model integration through Linked Data approach.

The research building is named “Vertigo”. It is the department building of the Built Environment faculty in Eindhoven University of Technology (TU/e). Vertigo has many experiment sites among the building. For example, there are several sensor sites in Vertigo that install sensors for different education and research uses. In this case study, Vertigo floor6 is selected as the research object, since there are several rooms erected with various sensors.

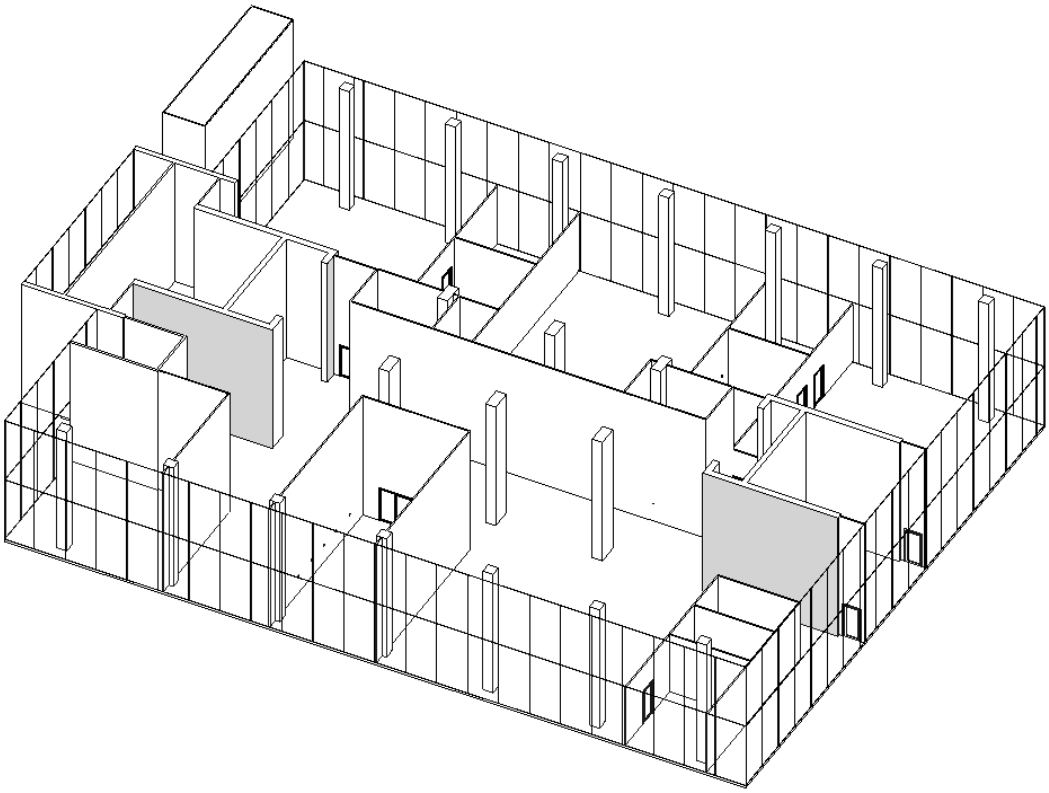


Figure 3 Vertigo floor6 model

The overall developing process for this case study is shown in figure 4. More detailed descriptions on the process will be displayed one by one, following the structure of the methodology chapter.

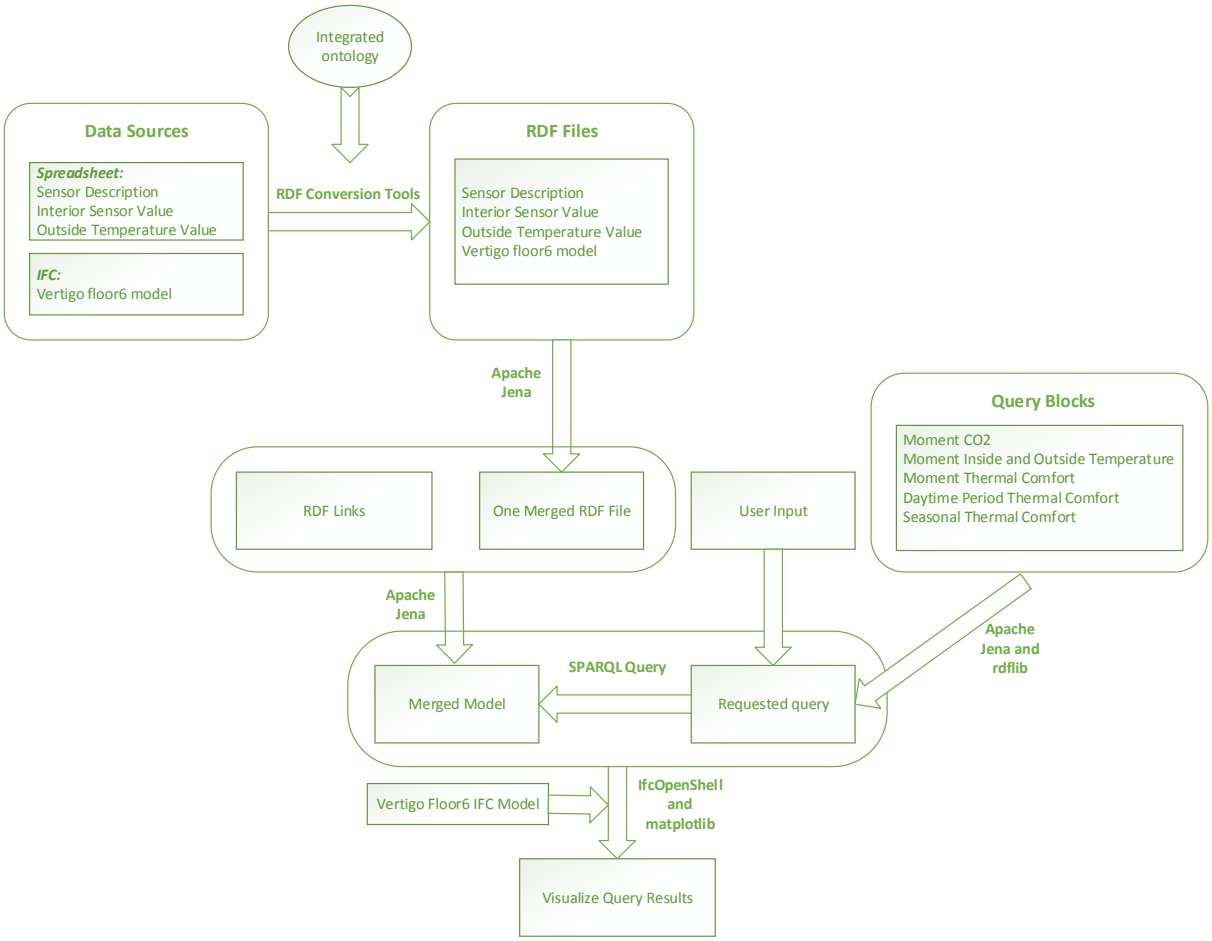


Figure 4 Case study process

5.1 Data collection

For this Vertigo Floor6 case, it is designed to help facility managers to manage different room environments. So different room environment performances are assumed to be the user information demand for facility managers. That means sensor data reflexes different room environment attributes and a BIM model are needed.

5.1.1 Data sources selection

There are five rooms in Vertigo floor6 that have installed sensors. The initial total number of sensors in the five rooms was 67, but some of them were moved to some other places, some needed repair, some sensors' measuring items do not meet the facility managers' information needs, and some measure the same value with different locations in a room. Since the sensor data selection criterion is reflecting different room environments, the sensor monitoring the same environment attributes in one room will not be repeated chosen. Apart from problematic sensors (no more work or has continuity issue) and sensors measuring the data that is not needed in this case study, there are only three rooms with 11 interior sensors left. The rooms and related sensor measuring items are shown in figure 5, under the category of "interior sensor values".

Since there is available interior temperature sensor data in the three selected rooms, for the facility managers' management convenience, the outside ambient temperature data is also collected in this case. The ambient temperature sensor is erected on the roof of Vertigo. As a result, facility managers can monitor inside and outside temperature differences.

What's more, to help facility managers understanding more about each sensor and to make the aspects of information available to query more abundant, a sensor description file is selected as supplementary data for those sensor values.

Then find a BIM model to perform the building space attributes for sensor data. Since there is no existing BIM model for Vertigo, a Revit model for Vertigo floor6 is created based on a 2D floor plan for Vertigo floor6 that is published on TU/e's official website.

To conclude, all the data sources for the data integration process are listed in figure 5.

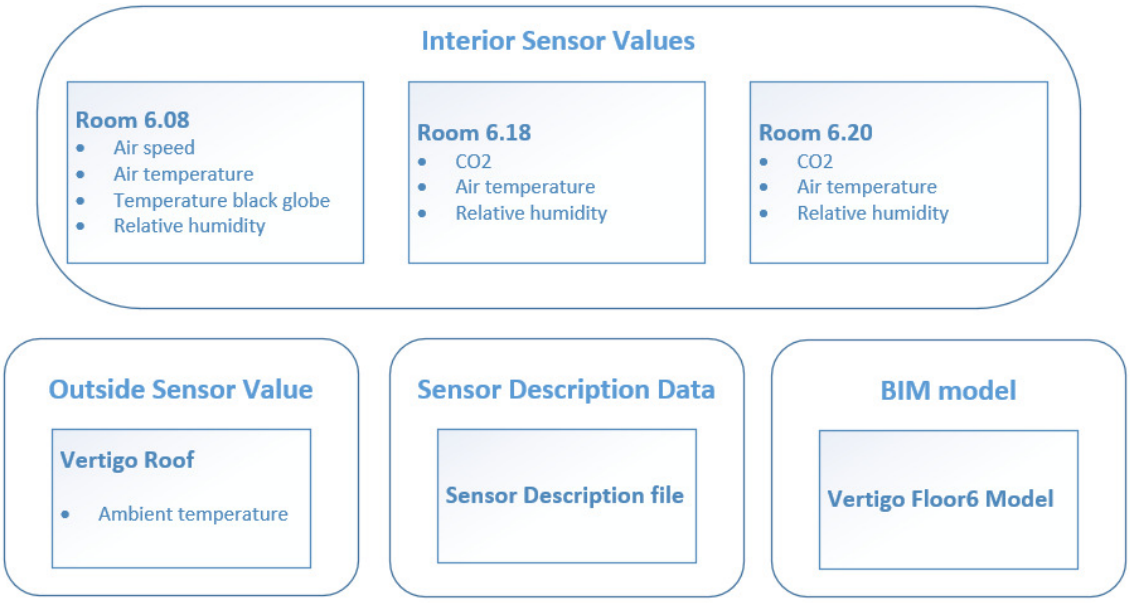


Figure 5 Data sources

5.1.2 Obtaining data access

The interior sensor values are loaded by Vertigo's facility managers into a Postgres database, and it can be viewed in the pgAdmin tool to help analysing data structure, as figure 6 shows. Jupyter Notebook is used to download SQL selected sensor data from the Postgress database, as figure 7 shows. A python coding example aiming at downloading an interior sensor's data in three time instants: 09:30, 13:00 and 17:30 for the whole 2015 year through Jupyter Notebook is shown in appendix A. The sensor data is saved as a local CSV file. The reason to select these three time instants is that they are representative for the time periods that people come to Vertigo and start work, people leave floor6 for lunch and people start leaving Vertigo after a whole day's work. These three moments can represent floor6's building environment in different statuses.

The interior sensors are first installed for education uses, but they are also opened to the students from the Built Environment Department. Students are provided with the database server's IP address, database name, student login name and password. Since in this case study, the sensor data is not planned to be published on the Internet, no further license needs to be applied for use.

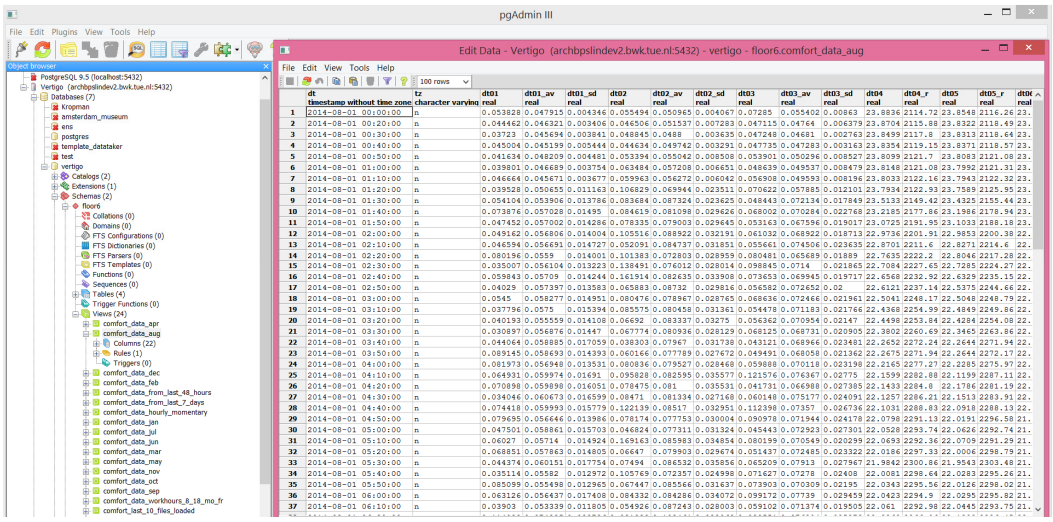


Figure 6 pgAdmin

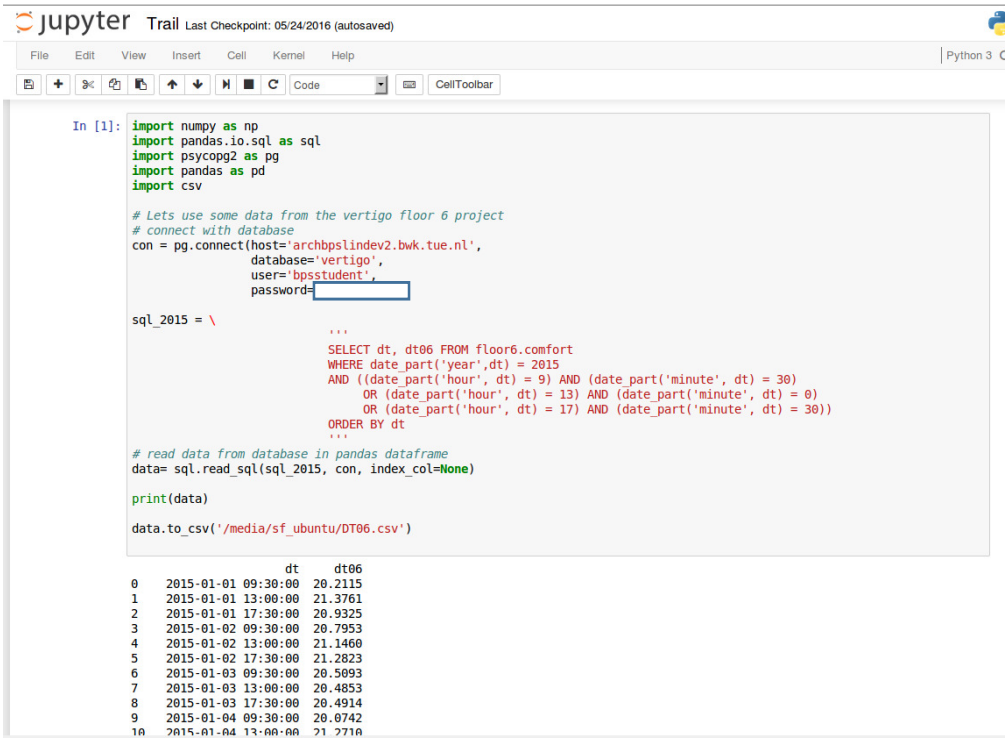


Figure 7 Jupyter Notebook

The outside temperature values are collected from the sensor installed on the roof of Vertigo. These data values are stored in the Solar (and weather) Measurement Station of SolarBEAT, and can be downloaded as a CSV file through the Data Retrieval Tool of the SEAC SolarBEAT Server. TU/e provides login code of the server to its students and employees. Without

publishing the data values on the Internet, there is no other license-related issue for using the data as well. Figure 8 shows a sample of the data monitoring window of the server, and figure 9 shows a sample of the window that exports data to a CSV file.

There are totally 48 sensors in the roof project, and a sensor named “T_amb_avg” is picked up from these sensors as a data source for data integration. This sensor measures the average ambient temperature outside Vertigo every 1 minute. The exported CSV file for the sensor values is processed in Excel, to select only the sensor values at the three time instants: 09:30, 13:00, 17:30 in every day of 2015, in order to keep pace with the interior sensor values and make the data size smaller.

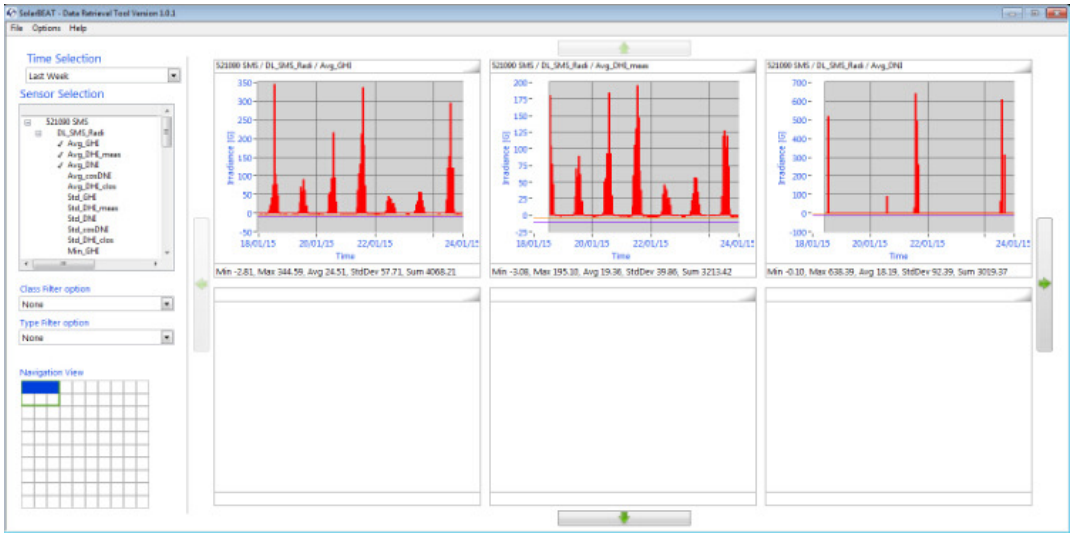


Figure 8 SAEC SolarBEAT monitoring

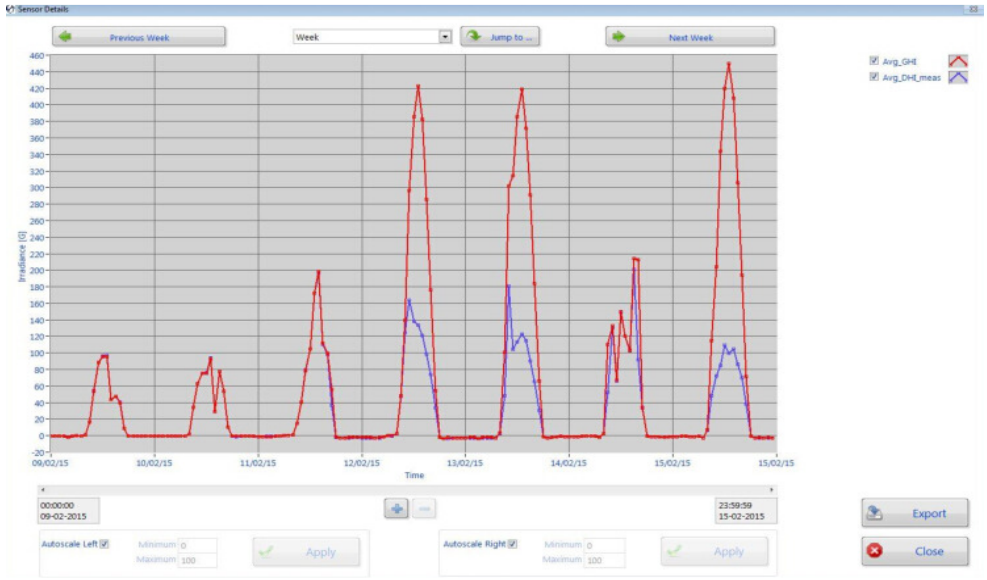
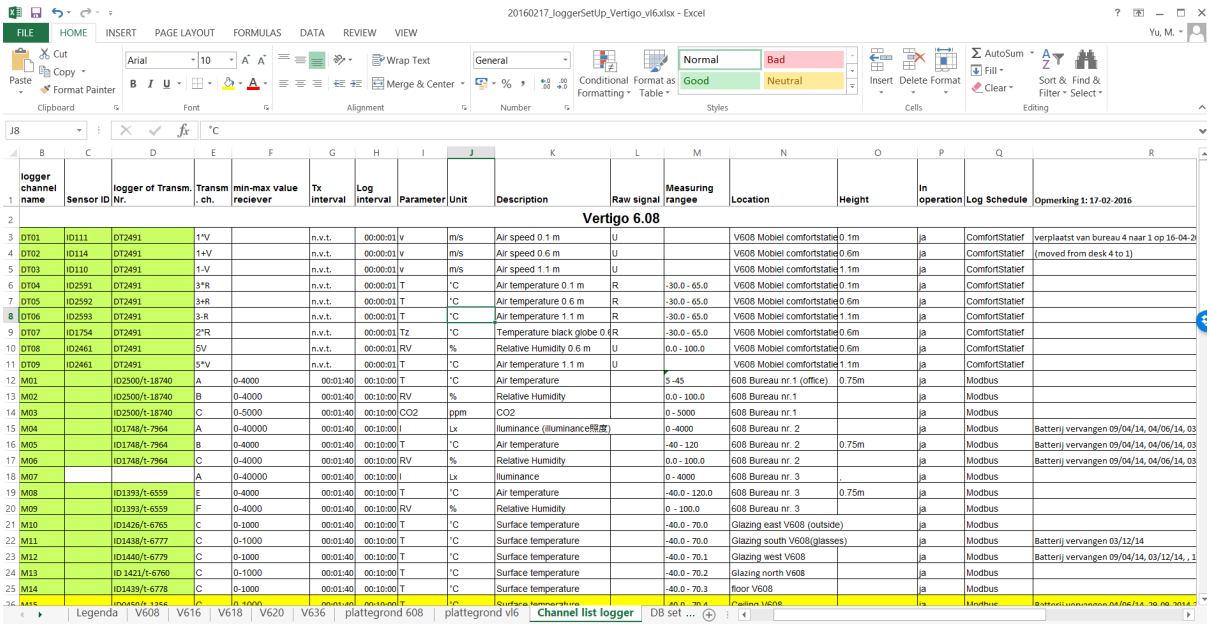


Figure 9 Export data

The sensor description file is generated from the sensor manager’s set-up records, as figure 10 shows. The set-up Excel file has plenty of records on sensors’ descriptive data. The sensor description table for the data integration extracts some columns from the set-up records so as to provide necessary information for facility managers to have a quick sensor background

check, as figure 11 shows. Also, the outside temperature sensor is added to the description table to make it a complete sensor background guide for facility managers.



logger channel name	Sensor ID	logger of Transm. Nr.	Transm. ch.	min-max value reciever	Tx interval	Log interval	Parameter	Unit	Description	Raw signal	Measuring rangee	Location	Height	In operation	Log Schedule	Opmerking 1: 17-02-2016
									Vertigo 6.08							
DT01	ID111	DT2491	1*V		n.v.t.	00:00:01	v	m/s	Air speed 0.1 m	U		V608 Mobiel comfortstatie	0.1m	ja	ComfortStatief	verplaatst van bureau 4 naar 1 op 16-04-2
DT02	ID114	DT2491	1+V		n.v.t.	00:00:01	v	m/s	Air speed 0.6 m	U		V608 Mobiel comfortstatie	0.6m	ja	ComfortStatief	(moved from desk 4 to 1)
DT03	ID110	DT2491	1-V		n.v.t.	00:00:01	v	m/s	Air speed 1.1 m	U		V608 Mobiel comfortstatie	1.1m	ja	ComfortStatief	
DT04	ID2591	DT2491	3*R		n.v.t.	00:00:01	T	°C	Air temperature 0.1 m	R	-30.0 - 65.0	V608 Mobiel comfortstatie	0.1m	ja	ComfortStatief	
DT05	ID2592	DT2491	3+R		n.v.t.	00:00:01	T	°C	Air temperature 0.6 m	R	-30.0 - 65.0	V608 Mobiel comfortstatie	0.6m	ja	ComfortStatief	
DT06	ID2593	DT2491	3-R		n.v.t.	00:00:01	T	°C	Air temperature 1.1 m	R	-30.0 - 65.0	V608 Mobiel comfortstatie	1.1m	ja	ComfortStatief	
DT07	ID1754	DT2491	2*R		n.v.t.	00:00:01	Tz	°C	Temperature black globe 0.6	R	-30.0 - 65.0	V608 Mobiel comfortstatie	0.6m	ja	ComfortStatief	
DT08	ID2461	DT2491	5V		n.v.t.	00:00:01	RV	%	Relative Humidity 0.6 m	U	0.0 - 100.0	V608 Mobiel comfortstatie	0.6m	ja	ComfortStatief	
DT09	ID2461	DT2491	5*V		n.v.t.	00:00:01	T	°C	Air temperature 1.1 m	U		V608 Mobiel comfortstatie	1.1m	ja	ComfortStatief	
M01		ID2500/t-18740	A	0-4000	00:01:40	00:10:00	T	°C	Air temperature		5 -45	608 Bureau nr.1 (office)	0.75m	ja	Modbus	
M02		ID2500/t-18740	B	0-4000	00:01:40	00:10:00	RV	%	Relative Humidity		0.0 - 100.0	608 Bureau nr.1		ja	Modbus	
M03		ID2500/t-18740	C	0-5000	00:01:40	00:10:00	CO2	ppm	CO2		0 - 5000	608 Bureau nr.1		ja	Modbus	
M04		ID1748/t-7964	A	0-40000	00:01:40	00:10:00	I	Lx	Iluminance (illuminance照度)		0 -4000	608 Bureau nr. 2		ja	Modbus	Batterij vervangen 09/04/14, 04/06/14, 03
M05		ID1748/t-7964	B	0-4000	00:01:40	00:10:00	T	°C	Air temperature		-40 - 120	608 Bureau nr. 2	0.75m	ja	Modbus	Batterij vervangen 09/04/14, 04/06/14, 03
M06		ID1748/t-7964	C	0-4000	00:01:40	00:10:00	RV	%	Relative Humidity		0.0 - 100.0	608 Bureau nr. 2		ja	Modbus	Batterij vervangen 09/04/14, 04/06/14, 03
M07			A	0-40000	00:01:40	00:10:00	I	Lx	Iluminance		0 - 4000	608 Bureau nr. 3		ja	Modbus	
M08		ID1393/t-6559	E	0-4000	00:01:40	00:10:00	T	°C	Air temperature		-40.0 - 120.0	608 Bureau nr. 3	0.75m	ja	Modbus	
M09		ID1393/t-6559	F	0-4000	00:01:40	00:10:00	RV	%	Relative Humidity		0 - 100.0	608 Bureau nr. 3		ja	Modbus	
M10		ID1426/t-6765	C	0-1000	00:01:40	00:10:00	T	°C	Surface temperature		-40.0 - 70.0	Glazing east V608 (outside)		ja	Modbus	
M11		ID1438/t-6777	C	0-1000	00:01:40	00:10:00	T	°C	Surface temperature		-40.0 - 70.0	Glazing south V608(glasses)		ja	Modbus	Batterij vervangen 03/12/14
M12		ID1440/t-6779	C	0-1000	00:01:40	00:10:00	T	°C	Surface temperature		-40.0 - 70.1	Glazing west V608		ja	Modbus	Batterij vervangen 09/04/14, 03/12/14, , 1
M13		ID 1421/t-6760	C	0-1000	00:01:40	00:10:00	T	°C	Surface temperature		-40.0 - 70.2	Glazing north V608		ja	Modbus	
M14		ID1439/t-6778	C	0-1000	00:01:40	00:10:00	T	°C	Surface temperature		-40.0 - 70.3	floor V608		ja	Modbus	

Figure 10 Sensor set-up file

Sensor Name	Log Interval	Unit	Description	Measurement Range	Location	Height	Log Schedule
DT03	00:00:01	m/s	Air speed at 1.1 m		Room 6.08	1.1m	ComfortStatief
DT06	00:00:01	˚C	Air temperature at 1.1 m	-30.0 - 65.0	Room 6.08	1.1m	ComfortStatief
DT07	00:00:01	˚C	Temperature black globe at 0.6 m	-30.0 - 65.0	Room 6.08	0.6m	ComfortStatief
DT08	00:00:01	%	Relative Humidity at 0.6 m	0.0 - 100.0	Room 6.08	0.6m	ComfortStatief
M03	00:10:00	ppm	CO2	0 - 5000	Room 6.08		Modbus
M38	00:10:00	˚C	Temperature	5 - 45	Room 6.18	0.75m	Modbus
M39	00:10:00	%	Relatieve humidity	0.0 - 100.0	Room 6.18		Modbus
M40	00:10:00	ppm	CO2	0 - 5000	Room 6.18		Modbus
M51	00:10:00	˚C	Temperature	-30.0 - 65.0	Room 6.20	0.75m	Modbus
M52	00:10:00	%	Relatieve humidity	0.0 - 100.0	Room 6.20		Modbus
M53	00:10:00	ppm	CO2	0 - 5000	Room 6.20		Modbus
T_amb_avg	00:01:00	˚C	Outside temperature		Vertigo roof		DL_SMS_Mete

Figure 11 Sensor description file

The BIM model for Vertigo floor6 is created through Revit for the data integration use. It is designed based on the official 2D floor plan for floor6. In fact there are some partial areas in floor6 with double-floor. Since there is no selected interior sensor installed on the upper-floor of floor6, to make the rooms installed with sensors clear to see, no upper-floor is drawn in the Revit model. Meanwhile, a small room near the floor6 structure is created as the outside space. This is for the convenience of inside and outside temperature visualization section. The three rooms and the outside space are tagged with room names through Revit, so that they will be allocated with more IFC space attributes that will give favours in the query process. The BIM model is shown in figure12.

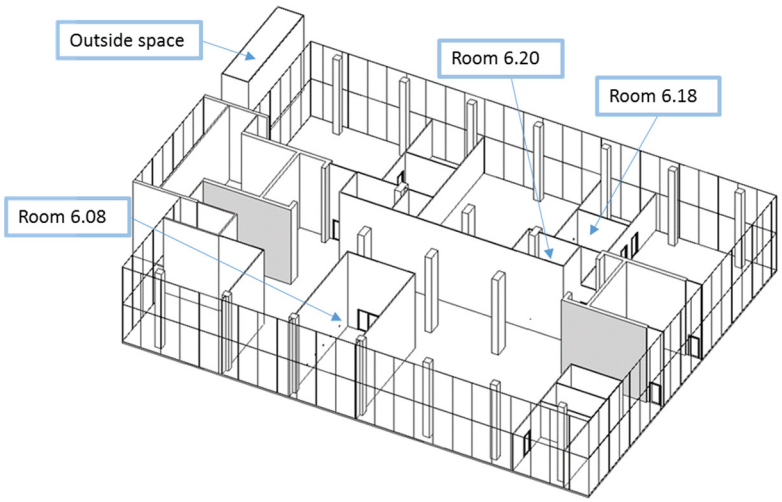


Figure 12 Rooms in floor6

5.2 Data analysis and conversion

The analysing data and converting data sections are combined together in this case study. These two processes are coordinated for the aim of creating a well merged RDF model.

5.2.1 *Resources naming strategy definition*

The basic form of URIs that was chosen for this case study is "hash URI". This kind of URIs contains a fragment that separates the rest of the URI with a hash character "#". The reason for choosing this URI form is that one of the following used RDF conversion tools, Google Refine, only supports hash URI.

The naming path for the interior sensor data begins with "http://Vertigo/floor6/", pluses unique sensor abbreviations and a "#" character, and then continues with the items to be described. The naming path for the outside temperature sensor data begins with "http://Vertigo/outsideTemperature#", and continues with the items to be described. The general sensor description data begins with "http://www.tue.nl/sensor#", and also concatenates with described items. The resources and properties related to BIM model will be named automatically by the RDF transformation tool. These resource and property names are only used as unique identifiers. As this use case does not contain Linked Data publishing process, the resources and properties do not need to be named under persistent uniform resource locator.

5.2.2 *Ontology development*

Since the theme of the collected data is about building and sensors, the developed ontology should also focus on describing the entities and relationships between the building elements and sensor data. Smart Appliances REFerence(SAREF) ontology (Smart Appliances, 2013) is a shared model of consensus that facilitates the match of existing assets in the smart appliances domain, and provides building-related classes to further expand model relationships. The SAREF ontology also imports an ontologies to explain geo space related entities: WGS84 Geo Positioning ontology (W3C Consortium, 2011). Meanwhile, IfcOWL ontology is used to

describe IFC-based vocabularies. To conclude, including the basic OWL, there are four existing ontologies applied in the case study's ontology for BIM and sensor data integration, as figure 13 shows.

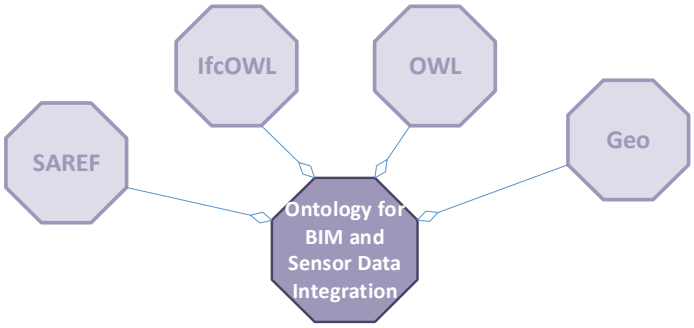


Figure 13 Model ontology composition

In addition, some of the paths that SAREF ontology was designed to describe sensor values and sensor descriptive data are distributed and winding, which does not suit our case well. To overcome the obstacle, this case study completes the ontology for BIM and sensor integration through creating new classes and properties related to sensor values and sensor description data. For example, class *sensorDescription:sensorName* is added to the ontology to introduce different sensor names in the sensor description file. Class *sensor:sensorRepresentation* is added to the ontology to represent different sensors used in this case. Property *sensor:isSensingValueOf* represents the belonging relationship between the sensor and the sensor value.

Figure 14 is created to shown the main part of the united ontology. The straight arrow represents the relationship between classes, and the dash arrow shows the relationships between the class and the individual. The blue, green and purple colours represent different description contents in the ontology: the sensor description part, the sensor value part and building model part.

Moreover, a file describing RDF ontology schema for self-created classes, individuals and properties is put in the appendix B.

5.2.3 *Data sources transformation and merging*

The RDF serialization used in this case is Turtle, because of its readable attribute. Since the data sources' original formats in this case are heterogenous, three transformation tools are selected to perform the Turtle transformation process. The detailed BPMN graph is shown in figure 15.

The sensor description file and the outside sensor values are first transferred from XLSX format into CSV format through Excel. The reason for this transform is that the next transformation tool Google Refine will convert the time cells in XLSX format and CSV format into different expression means. So first unify the spreadsheet format to avoid expression differences.

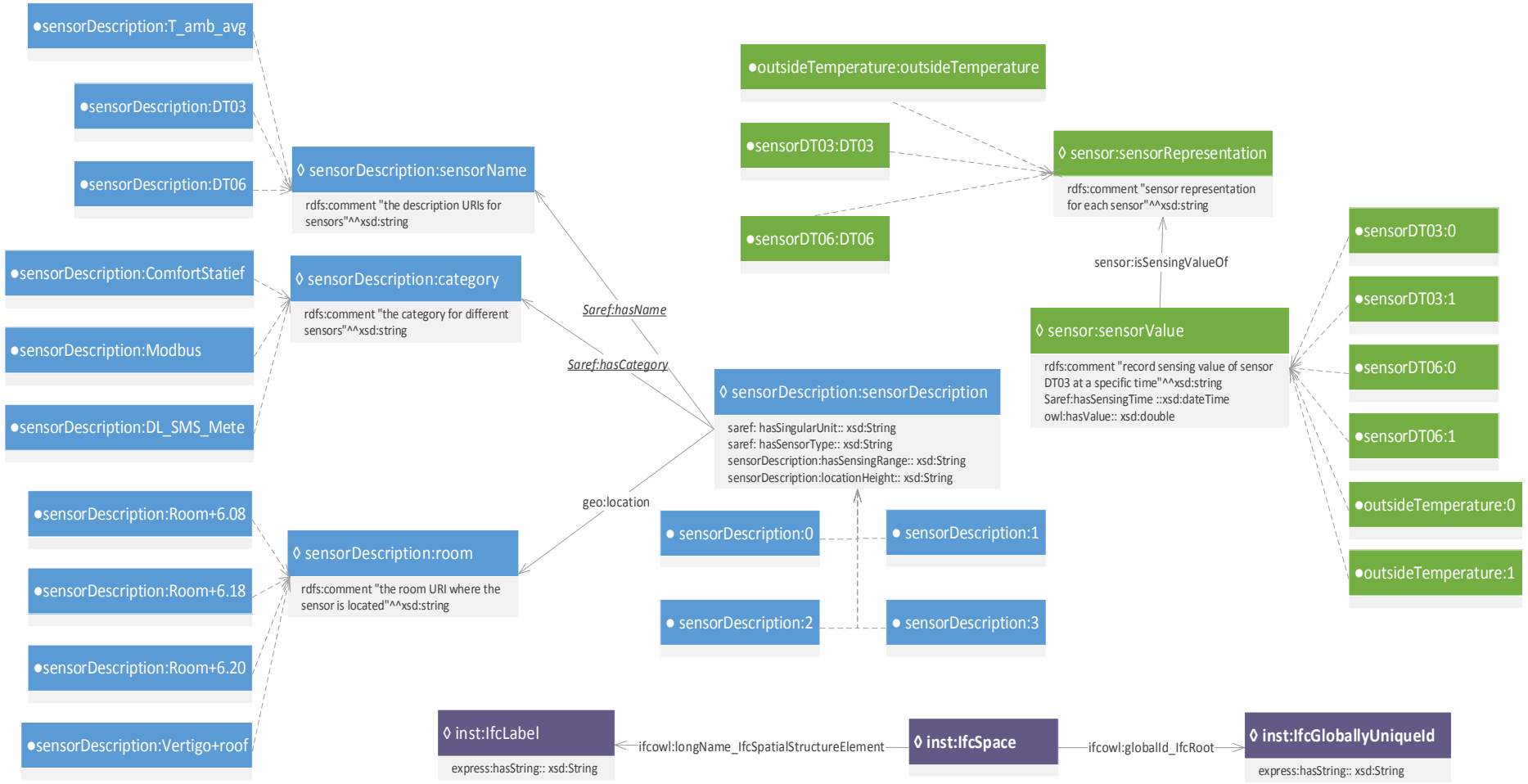


Figure 14 ontology before linking process

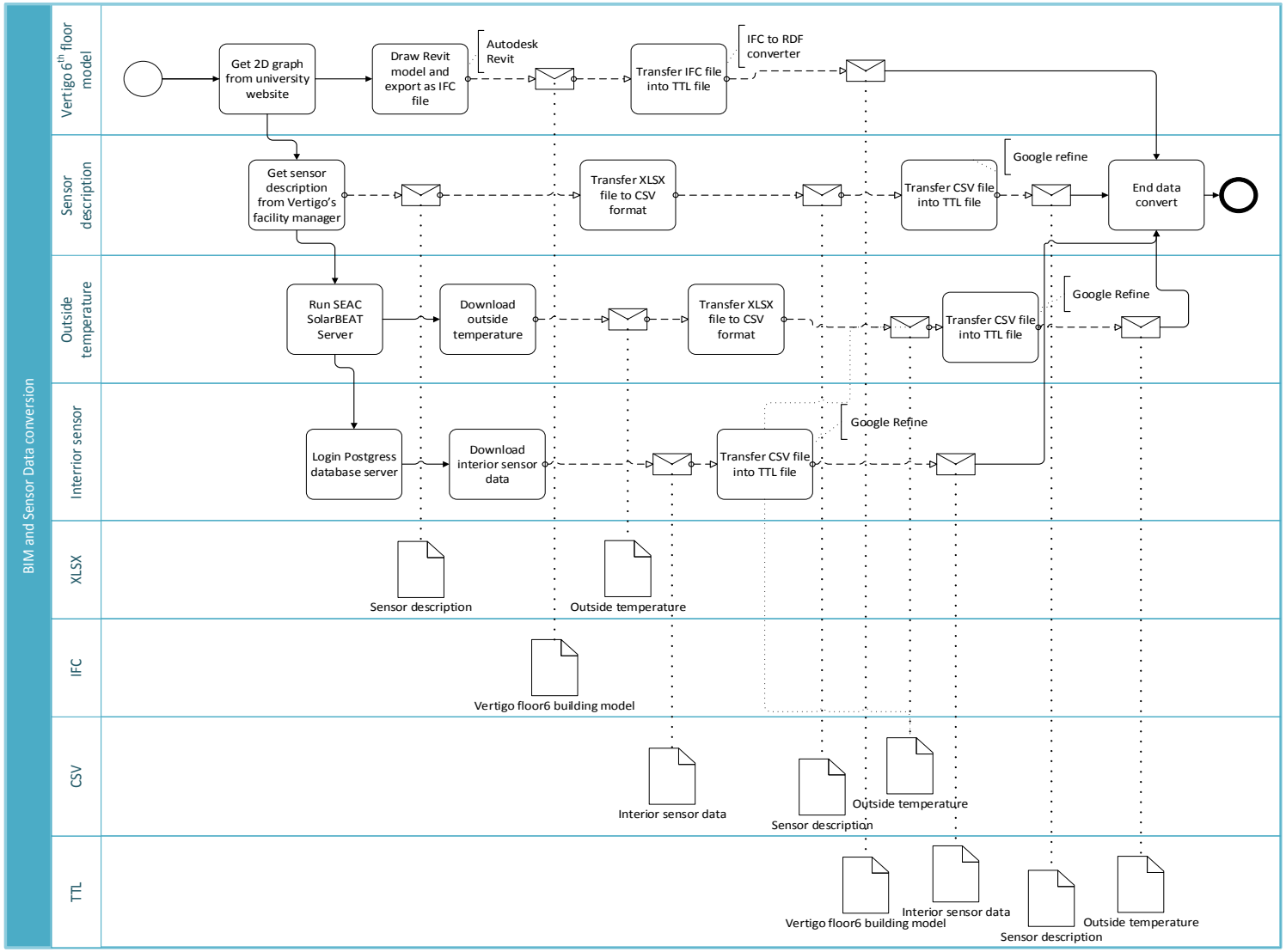


Figure 15 RDF transformation BPMN

Afterwards, Google Refine is used to transfer the sensor description file, the interior and outside sensor values from the CSV format into Turtle format. During employing Google Refine, the naming strategy and the developed ontology are used to guide the conversion, as figure 16 shows. The transformed Turtle file submits to the developed resource naming strategy and the integrated ontology.

The Revit model is exported as an IFC file and turns into a Turtle file through the IFC to RDF Converter.

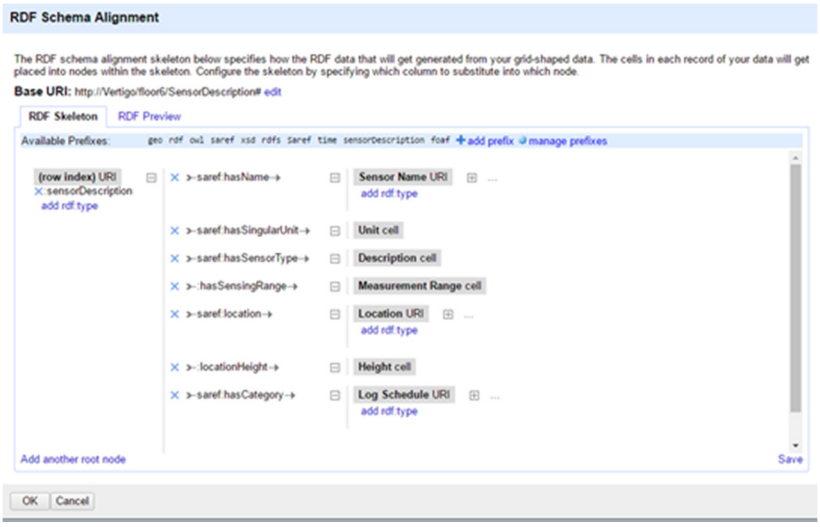


Figure 16 Google Refine transformation

After all the data sources are converted into Turtle files, next is to merge these files. The tool used for the data merging is a Java API named Jena RDF. Jena RDF API provides RDF writing, reading, simple query and other Java-based methods. In the merging data section, Jena RDF API is used to create a model which combines all the RDF data in it. The Java coding part can be viewed in the appendix C.

5.3 Data linking

Even though the merged model is created, the data from different sources are still isolated. No path is available to navigate through different information islands. The Linked Data approach aims at connecting related data that has not been linked previously, to help people exploring connected information world. The tool for this data linking process is Apache Jena API.

As the ontology development section shows, there are three parts of data in the merged model: the sensor description part, the sensor value part and the building model part. In the linking process, the sensor description part is settled as the middle part to generate links to the other two parts, as the sensor description part contains resources that has relationships with the sensor value part and the building information part.

The sensor description part has a “sensorDescription:room” class expressing the space location in string format, which can make link with “IfcLabel” class in the building model part. Since the “IfcLabel” class also aims at expressing room’s name string. The “sensor description” class in the sensor description part introduces each sensor’s basic descriptive information, which is correspondent with “sensor representation” class in the sensor value part that displays every sensor value’s sensor belongingness. These two connections link the three separated information part as one resources-interrelated model. This makes full preparation for the data query process.

The ontology graph with linked relationships is shown in figure 17. And the corresponding linking properties’ schema is added to appendix D. The Java coding to achieve the links based on Apache Jena API is also put in appendix C.

5.4 Query blocks development

5.4.1 User demands analysis and query topics selection

The sensor data collected for integration is about interior relative humidity, interior CO2 level, interior air speed, interior and outside air temperature and black globe temperature. All these measuring items have their own meanings to help facility managers monitoring building environment.

High CO2 level in a room can influent people’s health and working efficiency. Monitoring building CO2 level and adjusting indoor ventilation according the monitoring data contribute to create a healthy and comfortable working environment. So, it is helpful for facility managers to analyse CO2 level in each room and make suitable reactions to the rooms that CO2 level frequently goes high.

Even though Vertigo’s indoor temperature is nearly stable, it is still useful for facility managers and people in the building to be aware of inside and outside temperature differences. This can give instructions for people on clothing wearing inside and outside office.

In Room 6.08, there are four kinds of sensors measuring values that are core variables for calculating human thermal comfort: relative humidity, air speed, air temperature, and black globe temperature. According to ANSI/ASHRAE Standard 55-2010, thermal comfort is defined as “condition of mind which expresses satisfaction with the thermal environment and is assessed by subjective evaluation” (ANSI/ASHRAE Standard 55, 2010). This index can be used to reflect people’s thermal comfort feeling about the working space. The Predicted Mean Vote (PMV) is recommended as an explicit definition of the thermal comfort zone by ISO Standard 7730 (ISO, 1984). And the Predicted Percentage of dissatisfied (PPD) is a function of PMV to indicate the human dissatisfied rate of a specific thermal environment. Facility managers could use the thermal comfort index to assess the indoor thermal environment. Although there is only one room with enough sensor values to calculate the thermal comfort, facility managers can treat the thermal comfort of this room in different time period as representations to analyse floor6’s space comfort performances.

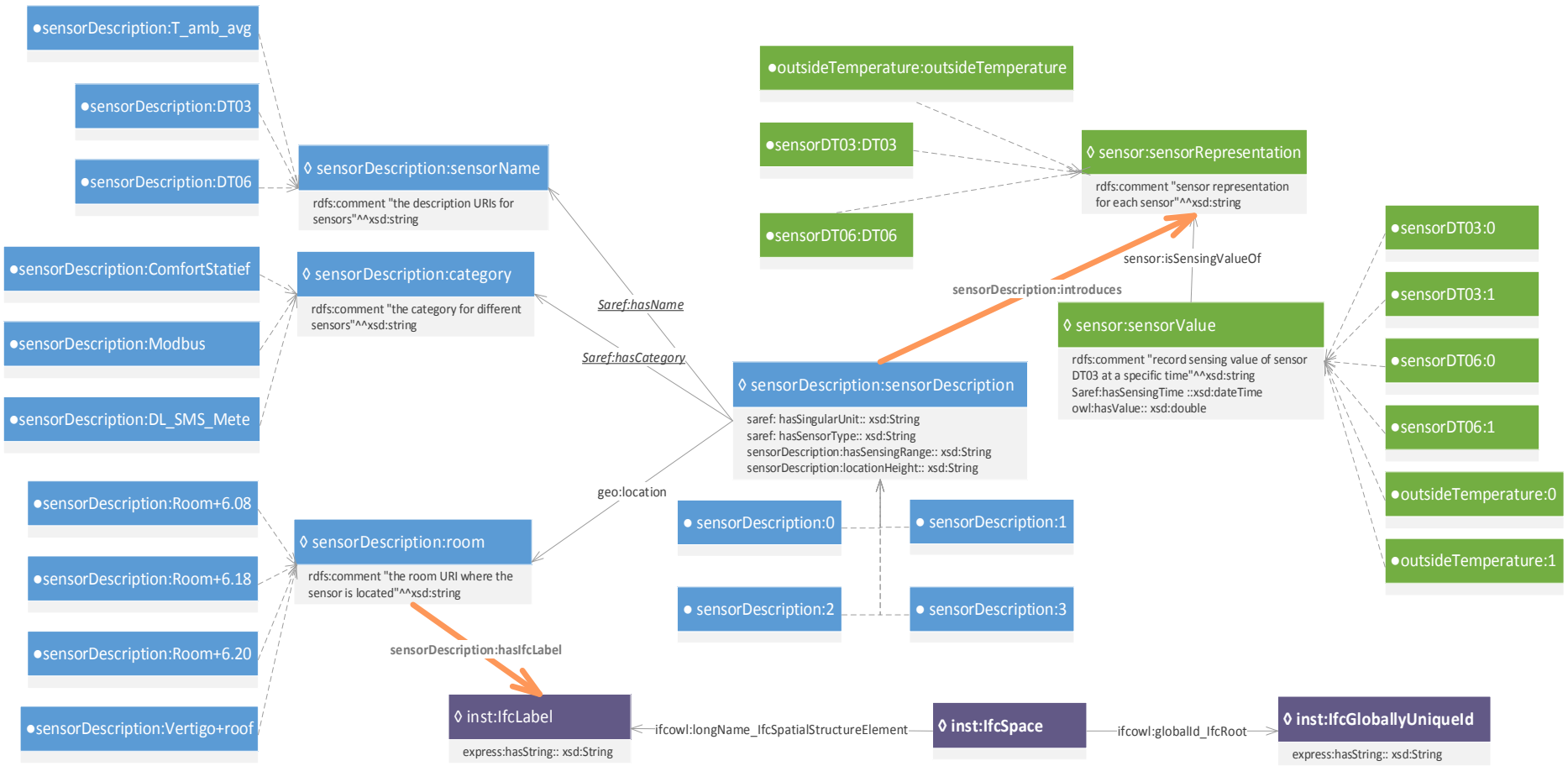


Figure 17 Ontology after linking process

After discussing the meanings of different sensor types for facility managers on building environment monitoring, concrete query topics could be chosen. In this case study, two categories with five types of querying topics are performed:

(1) Daily query for floor6's environment. This querying category focuses on providing the facility managers with Vertigo floor6's daily environment monitoring data. Facilty managers could choose any date in 2015 to retrieve the corresponding environment monitoring data. Among each date in 2015, there are three time instants provided as choices for the daily query time: 09:30, 13:00, and 17:30. The representations of the three time instants have been discussed in the previous section. This daily environment query helps facility managers freely choosing the date that they want to view monitoring data on, and could be helpful when the facility managers have special analysing demand on specific date in 2015. There are three detailed query topics under the daily query category:

- The momentary CO2 level query. This topic focuses on querying the CO2 levels at the selected moment for the three rooms in floor6: room 6.08, room 6.18, room 6.20.
- The momentary inside and outside temperature query. This topic focuses on querying the temperature values at the selected moment for the outside space and three interior rooms: room 6.08, room 6.18, room 6.20.
- The momentary thermal comfort query. This topic focuses on querying the thermal comfort value at the selected moment for room 6.08.

(2) Conclusive query for floor6's environment. This category aims at providing conclusive thermal environment analysis for facility managers. Some data processing is performed to achieve synthetical representations of floor6's thermal comfort performances during different time periods. There are two detailed query topics under the conclusive query category:

- The seasonal thermal comfort query. This topic is designed to calculate the average PMV values in month April, July, October and February, and to generate the corresponding PPD values of the four months. The PPD values of these four months are used to stand for people's dissatisfied rates about floor6's thermal environment in the four seansons: spring, summer, autumn and winter.
- The daytime period thermal comfort query. This topic is designed to calculate the average PMV values in April for the three time instants: 09:30, 13:00 and 17:30, and to generate the corresponding PPD values of the three moments. These three PPD values could represent floor6's thermal comfort distribution among different time periods in the day.

5.4.2 SPARQL blocks development

After decided the query topics for facility managers, the SPARQL query blocks should be developed. The SPARQL queries and further data process are tested both on Java Apache Jena API and Python rdflib. In the end, the moment CO2 level query block, the moment inside and outside temperature query block and the moment thermal comfort query block are developed under Python rdflib. The Python codes can be found in appendix E. The seasonal and daytime

period thermal comfort query blocks are developed on Apache Jena API. The Java code can be found in appendix C. More detailed descriptions for the blocks design are conducted below.

(1) The momentary CO2 level query block

This query block starts from inviting the facility managers to input the day that they want to query on (in 2015), and to select one of the three time instants: 09:30, 13:00, 17:30 to query at. Then the python program will execute a SPARQL query to find the CO2 levels at the selected time in the three rooms on floor6. The CO2 levels and their corresponding IfcGloballyUniqueIds will be printed out and stored for visualization use.

(2) The momentary inside and outside temperature query block

Like the moment CO2 level query block, this query begins at the facility managers inputing the day and the time that they want to query. Then the python program executes a SPARQL query to find the temperature values of the outside space and the three inside rooms at the selected moment. The temperature values and their corresponding *IfcGloballyUniqueIds* will be printed out and stored for visualization use.

(3) The momentary thermal comfort query block

Likewise, after facility managers inputing the day and time that they want to query at, the python program executes a SPARQL query to find the air temperature value, air speed value, relative humidity value and black globe temperature value of room6.08 at the selected moment. Then the thermal comfort values standed by PMV and PPD indexies are calculated. The equations of the calculation are shown in figure 18. Some of the parameters in the functions are presumed as constants according to the ASHRAE Standard. The equations are referenced from Fanger's thermal comfort model (Tech, 2002). The calculated PPD value and room6.08's IfcGloballyUniqueId will be printed out and stored for visualization use.

$$PMV = [0.303e^{-0.036M} + 0.028]\{(M-W) - 3.96E^{-8}f_{cl}[(t_{cl}+273)^4 - (t_r+273)^4] - f_{cl}h_c(t_{cl}-t_a) - 3.05[5.73 - 0.007(M-W) - p_a] - 0.42[(M-W) - 58.15] - 0.0173M(5.87 - p_a) - 0.0014M(34 - t_a)\}$$

With

$$f_{cl} = \frac{1.0 + 0.2I_{cl}}{1.05 + 0.1I_{cl}}$$

$$t_{cl} = 35.7 - 0.0275(M-W) - R_{cl}\{(M-W) - 3.05[5.73 - 0.007(M-W) - p_a] - 0.42[(M-W) - 58.15] - 0.0173M(5.87 - p_a) - 0.0014M(34 - t_a)\}$$

$$R_{cl} = 0.155I_{cl}$$

$$h_c = 12.1(V)^{1/2}$$

Where

e	Euler's number (2.718)
f_{cl}	clothing factor
h_c	convective heat transfer coefficient
I_{cl}	clothing insulation [clo]
M	metabolic rate [W/m^2] 115 for all scenarios
p_a	vapor pressure of air [kPa]
R_{cl}	clothing thermal insulation
t_a	air temperature [°C]
t_{cl}	surface temperature of clothing [°C]
t_r	mean radiant temperature [°C]
V	air velocity [m/s]
W	external work (assumed = 0)

Since PPD is a function of PMV, it can be defined as

$$PPD = 100 - 95e^{[-(0.3353PMV^4 + 0.2179PMV^2)]}$$

Figure 18 PMV and PPD equations

(4) The seasonal thermal comfort query block

This query block begins from using the SPARQL to query the everyday air speed value, air temperature value, relative humidity value and black globe temperature value during the four months at 13:00. Then calculate the average PMV value of each month based on these sensor values, and also calculate the corresponding PPD values of the four months to represent the seansonal thermal comfort statuses. The PPD values generated by Java programming are copied to the python program, in order to perform the data visualization together with the other query blocks through pycharm.

(5) The daytime period thermal comfort query block

This query block first performs the SAPRQL query to collect the four kinds of sensor values from room 6.08 at the three time instants during April for PMV value calculation. Through the average PMV values for April on the three time instants, corresponding PPD values representing the three time periods in the day are generated. The PPD values generated from Java programming are also copied to the python program for the following visualization use.

5.5 Data visualization

In this part, the interior moment CO2 levels, inside and outside moment temperature values and the interior moment thermal comfort values are visualized through IfcOpenshell. The seasonal thermal comfort values and the daytime period thermal comfort values are visualized by histograms through matplotlib[8]. In the IfcOpenShell visualization part, the abstract numeric values are divided into several value ranges and are presented by different colours. In the visualized building model generated by IfcOpenShell, the sensor values in different rooms are presented in colours that are used to on behalf of specific data ranges. Facility managers can get an intuitive impression about the floor's environment performances through different colours. The source codes for the visualization part can be viewed in appendix E.

(1) The momentary CO2 levels

Randomly picking up a time to view the interior CO2 levels, the CO2 levels in the three rooms are shown as figure 19. The CO2 level's ranges are assigned to different colours, and they are also shown in figure 19. From this figure, it is obvious that room6.18's CO2 concentration is at a high level and needs facility manager's attention.

[8] http://matplotlib.org/

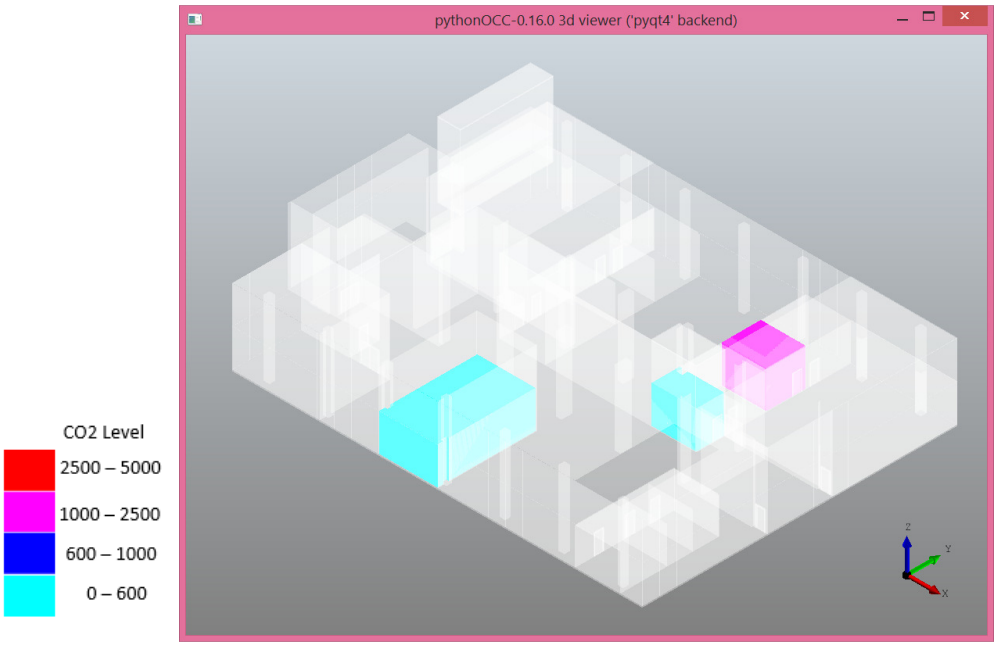


Figure 19 CO2 levels

(2) The momentary temperature

Randomly picking up a time to view the inside and outside temperature values, the temperature data in the three rooms and outside space is shown as figure 20. The temperature data ranges are assigned to different colours, and they are also shown aside. From the graph, it shows that room6.18's temperature is a bit high. The difference between inside and outside temperature is not big, which is a good news for facility managers on energy consumption.

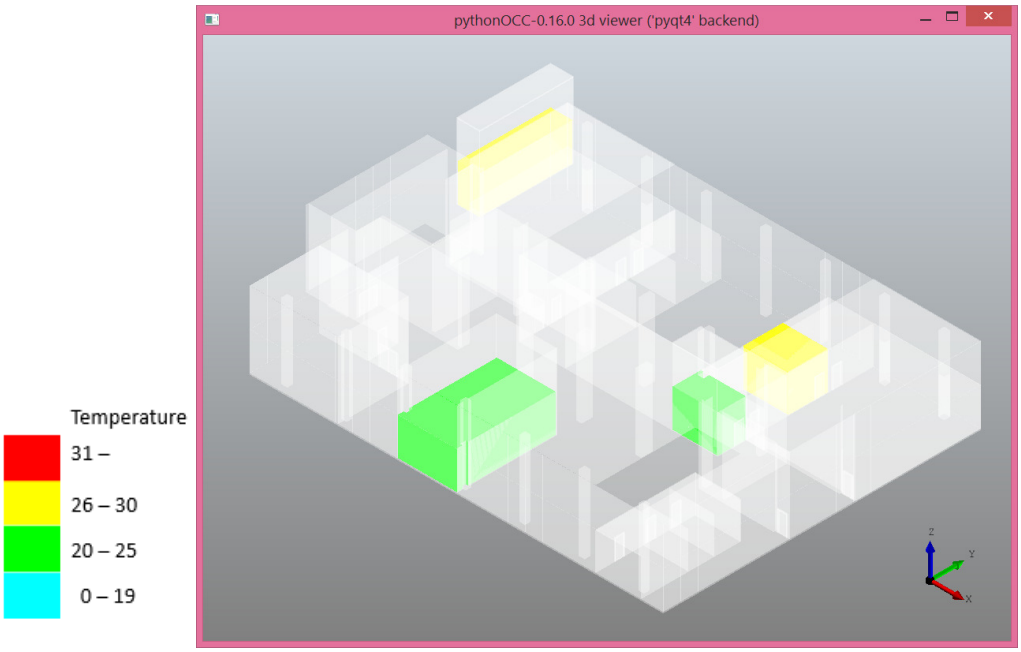


Figure 20 Inside and outside temperature

(3) The momentary thermal comfort

Also pick up a time for thermal comfort query. As figure 21 shows, the thermal comfort for room 6.08 at the query time is quite awful, above 75% of people are unsatisfied with the thermal environment. It can be deduced that the whole floor6's thermal environment performance is not satisfying. Facility managers could open more windows and increase ventilation rate to avoid this situation.

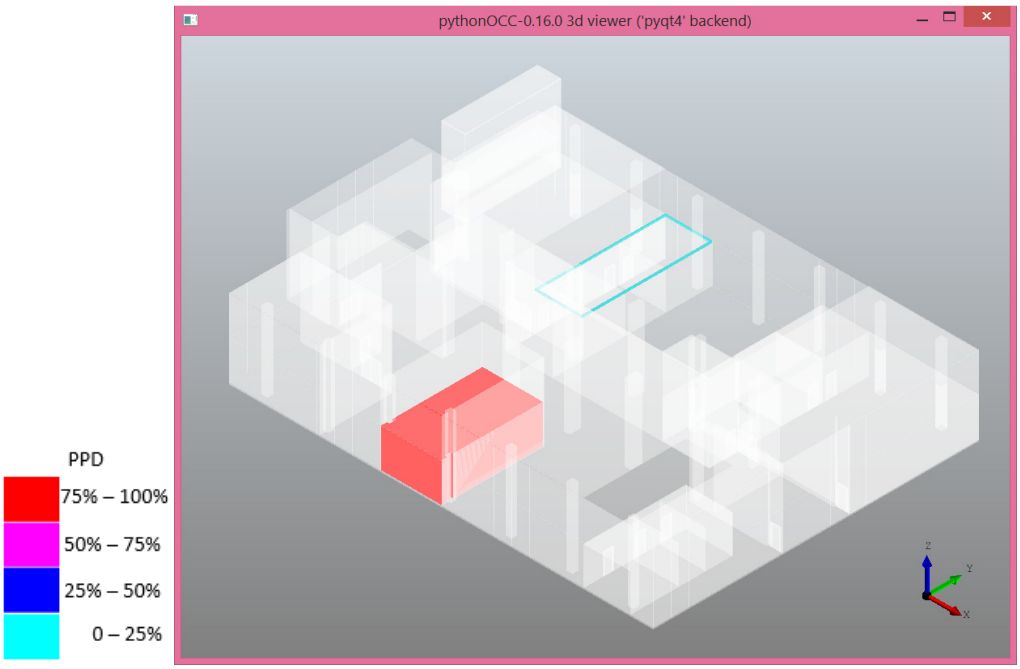


Figure 21 Moment thermal comfort

(4) The seasonal thermal comfort

Figure 22 shows the thermal comfort distribution of floor6 among the four months . From the figure, it can be deduced that summer is the season with the worst thermal comfort experience of Vertigo floor6, while autumn and spring are quite comfortable seasons for floor6. As a facility manager, building thermal environment management in summer should be paid additional attention.

(5) The daytime period thermal comfort

Figure 23 shows the thermal comfort distribution of floor6 at the three time instants. According to the results, the thermal environment in the morning period is not so satisfied. But the effect of ventilation system of floor6 is proved to be good on a certain sense, since the thermal comfort is getting better and better at lunch time and in the afternoon compared with the morning status. Facility managers could start the ventilation earlier in the morning to improve the thermal comfort experience.

```
The average predicted percentage of dissatisfied occupants
for February is 13.17%,
for April is 39.82%,
for July is 89.78%,
and for October is 11.25%.
```

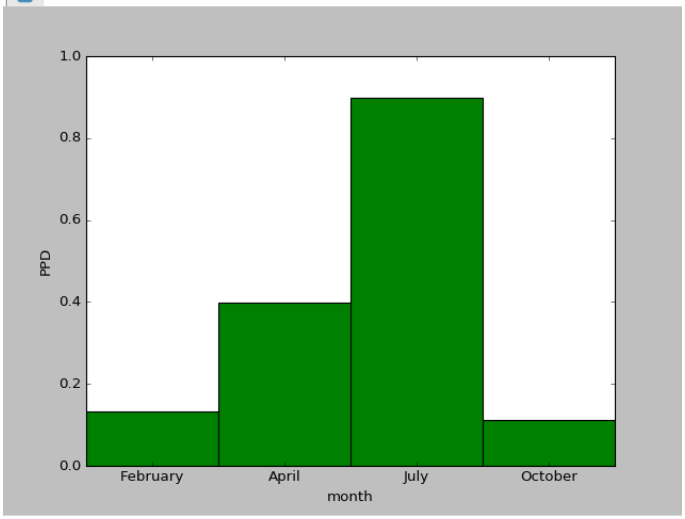


Figure 22 Seasonal thermal comfort

```
The average predicted percentage of dissatisfied occupants for April
 at 09:30 is 49.54%,
 at 13:00 is 39.82%,
 at 17:30 is 37.00%.
```

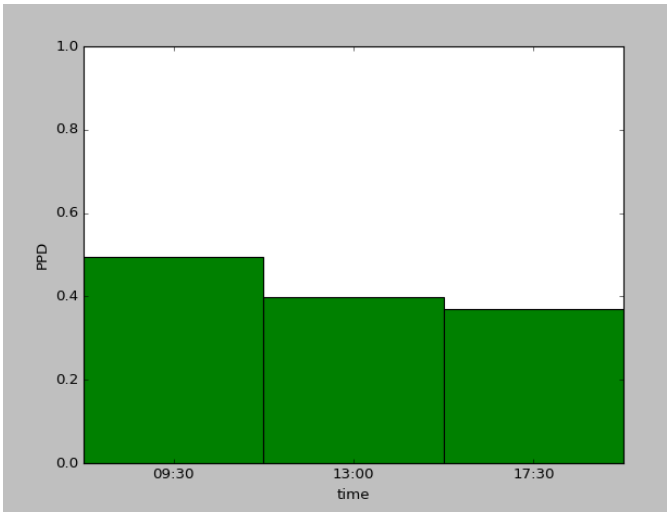


Figure 23 Daytime period thermal comfort

6. Conclusion

This thesis describes a methodology to achieve BIM model and sensor data integration by using Linked Data, and implements the method into a case study. The conclusion of this thesis starts from answering the research questions raised in the beginning of the thesis basing on the case study:

Sub question 1: "*How to collect and convert different data sets into RDF format for building performance analyses?*"

In the case study, there are plenty of sensor data sources that can be chosen for the data integration. The sensor data selection criteria are the availability, continuity and accuracy of the sensor data itself and the facility managers' information demand. For Vertigo, based on these two criteria, three categories of sensor values are chosen: the interior CO2 level for users' health and environment comfort monitoring, the interior and outside temperature monitoring for people's comfort feeling, and the thermal comfort index for thermal environment comfort monitoring. Moreover, in this case, a sensor description file collected from a facility manager is also selected, for the use of sensor background supplement. After collecting all the sensor data and a BIM model, the data format processing and converting to RDF expression are conducted. The conversion tools to turn the original data into RDF format are chosen according to different original data formats. All the conversion processes are conducted under the ontology's instruction which is developed for the integrated data. When all the files are converted into RDF format, the file merging is performed by RDF merging software.

Sub question 2: "*What kinds of links between BIM and sensor data can be created? How to achieve the linking?*"

In this case study, the sensor description part is selected as the middle part to generate connections to the data from other data sources. The resource links can be created when there are information overlap or logic relationship exists between the data resources.In Vertigo's case, there are two kinds of links created: the link focuses on the same room's name strings that are defined both in the sensor description part and the IFC model part, and the link focuses on describing the introduction relationship between the sensor description part and the sensor representation class from the sensor value part. These two types of links are generated through Apache Jena API by creating corresponding RDF triples that represent the links and adding them to the merged RDF model.

Sub question 3: "*How can the integrated data help facility managers analysing building performances?*"

In this case, the building performance analyses are conducted through the SPARQL query and the query results processing. Based on the three categories of sensor data, corresponding query topics are created that may be helpful for the facility managers to analyse the building

environment comfort. A python program is created to perform all the topic queries and data processing. The facility managers can freely ask the related building environment information through the topic and time inputs, and receive visualizable results from the query processing.

The main question: "*How to integrate a BIM model with sensor data to support facility managers in analysing building performances, by using Linked Data?*"

In summary, this can be achieved through several steps: 1) based on facility managers' analysing demand and selected sensor data, 2) transform all the selected sensor data and the BIM model into RDF files under a developed ontology, 3) merge these files and connect them through create links between different data sources, 4) develop SPARQL query or use other RDF processing methods to conduct the integrated information analyses for the building operation performance assessment.

In the case study, there are four different information sources from the sensor data and the BIM model that are needed to be processed together. It is proved that through using Linked Data, it is convenient to unify and query information coming from different data formats without information lost or distortion and interoperability problems. This information integration method can help facility managers synthesizing different domains and different formats information in order to comprehensively and quickly analysing the building operation performances.

Recommendations for further research:

Create links with automate software

In this case study, all the links are created through Java coding. This is fine when the number of links are limited. However, for the large-size projects, hundreds of links may be needed, and coding the links one by one is not efficient. Using a software that can automatically generate links based on the selected data sources is a more convenient way.

Visualization performance with annotation

In this case study, the query data visualization under the moment query category is only focused on the building model itself, and no literal annotation is created for the visualization. This may cause trouble for the facility managers in finding the corresponding rooms in the building model. For future research, some extensions on the IfcOpenShell could be developed to give the building performance visualization more annotative information.

Data processing within SPARQL

In this case study, the thermal comfort calculation is separated from the SPARQL query. But it may be more convenient to conduct the thermal comfort calculation and other possible data processes within the SPARQL query. Through this, the building performance analysing process

can be conducted on all the software that can support SPARQL and will not be limited only on software that has both SPARQL and further data processing abilities.

Ontology development

In this case study, four existing ontologies are used for generating the integrated ontology. However, for the SAREF and Geo ontologies, only several properties from them are implied into the integrated ontology. For the IfcOWL ontology, it only works for the BIM model part. And the OWL ontology is used for the general RDF description. To conclude, there is still no public admitted ontology for the BIM and sensor data integration use. The development of public recognized ontology for the BIM and sensor data integration is quite urgent for the integration method promotion and future research.

References

ASHRAE. (2010). ANSI/ASHRAE Standard 55-2010. *Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers*, Inc.

Azhar, S. (2011). Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and Management in Engineering*.

Azhar, S., Nadeem, A., Mok, J. Y., & Leung, B. H. (2008, August). Building Information Modeling (BIM): A new paradigm for visual interactive modeling and simulation for construction projects. In *Proc., First International Conference on Construction in Developing Countries* (pp. 435-446).

Barnaghi, P., Meissner, S., Presser, M., & Moessner, K. (2009). Sense and sens' ability: Semantic data modelling for sensor networks. In *Conference Proceedings of ICT Mobile Summit 2009*.

Becerik-Gerber, B., & Rice, S. (2010). The perceived value of building information modeling in the US building industry. *Journal of information technology in Construction*, *15*(2), 185-201.

Beetz, J., Van Leeuwen, J., & De Vries, B. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *23*(01), 89-101.

Berners-Lee, T. (2006). Linked Data. *https://www.w3.org/DesignIssues/LinkedData.html*

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web.*Scientific american*, *284*(5), 28-37.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web.*Scientific american*, *284*(5), 28-37.

Bizer, C., Heath, T., & Berners-Lee, T. (2009). Linked data-the story so far.*Semantic Services, Interoperability and Web Applications: Emerging Concepts*, 205-227.

Brickley, D., & Guha, R. V. (2000). Resource Description Framework (RDF) Schema Specification 1.0: W3C Candidate Recommendation 27 March 2000.

Cahill, B., Menzel, K., & Flynn, D. (2012). BIM as a centre piece for optimized building operation.

Corry, E. (2014). A semantic web approach to enable the holistic environmental and energy management of buildings (Doctoral dissertation).

Corry, E., Coakley, D., O'Donnell, J., Pauwels, P., & Keane, M. (2013). The role of linked data and the semantic web in building operation. In *13th annual International Conference for Enhanced Building Operations (ICEBO)*.

Curry, E., O'Donnell, J., Corry, E., Hasan, S., Keane, M., & O'Riain, S. (2013). Linking building data in the cloud: Integrating cross-domain building data using linked data. *Advanced Engineering Informatics*, *27*(2), 206-219.

Curry, E., O'Donnell, J., Corry, E., Hasan, S., Keane, M., & O'Riain, S. (2013). Linking building data in the cloud: Integrating cross-domain building data using linked data. *Advanced Engineering Informatics*, *27*(2), 206-219.

Decker, S., & Hauswirth, M. (2008, September). Enabling Networked Knowledge. In *CIA* (Vol. 8, pp. 1-15).

Eastman, C., Eastman, C. M., Teicholz, P., & Sacks, R. (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*. John Wiley & Sons.

Eid, M., Liscano, R., & El Saddik, A. (2007, June). A universal ontology for sensor networks data. In *Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on* (pp. 59-62). IEEE.

Enkovaara, E., Salmi, M., & Sarja, A. (1988). *RATAS project: computer aided design for construction*. Building Book Limited.

Fischer, M., & Kam, C. (2002). PM4D Final Report€, CIFE Technical Report (TR143). *CIFE, Stanford University*.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2), 199-220.

HM Goverment. (2012). Industrial strategy: government and industry in partnership. *https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/34710/12-1327-building-information-modelling.pdf/*.

ISO, I. (1984). Standard 7730. Moderate thermal environments-determination of the PMV and PPD indices and specification of the conditions for thermal comfort. *Geneva: International Standards Organization*.

Janowicz, K., *Bröring,* A., Stasch, C., & Everding, T. (2010, September). Towards meaningful uris for linked sensor data. In *Towards digital earth: Search, discover and share geospatial data. Workshop at Future Internet Symposium*.

Janowicz, K., Bröring, A., Stasch, C., & Everding, T. (2010, September). Towards meaningful uris for linked sensor data. In *Towards digital earth: Search, discover and share geospatial data. Workshop at Future Internet Symposium*.

Kim, J. H., Kwon, H., Kim, D. H., Kwak, H. Y., & Lee, S. J. (2008, May). Building a service-oriented ontology for wireless sensor networks. In*Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on* (pp. 649-654). IEEE.

Korpela, J., Miettinen, R., Salmikivi, T., & Ihalainen, J. (2015). The challenges and potentials of utilizing building information modelling in facility management: the case of the Center for Properties and Facilities of the University of Helsinki. *Construction Management and Economics*, *33*(1), 3-17.

Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Karstila, K., & Wix, J. (2006). Industry Foundation Classes IFC2× 3. *International Alliance for Interoperability*, 467-476.

Newton, R. S. (2004). Inadequate interoperability in construction wastes 415.8 billion annually. *AECNews. com*, *13*.

O'Flynn, B., Jafer, E., Spinar, R., Keane, M., Costa, A., Pesch, D., & O'Mathuna, C. (2010, September). Development of miniaturized wireless sensor nodes suitable for building energy management and modelling. In*eWork and eBusiness in Architecture, Engineering and Construction: Proceedings of the European Conference on Product and Process Modelling 2010, Cork, Republic of Ireland, 14-16 September 2010* (p. 253). CRC Press.

Patel-Schneider, P. F., Hayes, P., & Horrocks, I. (2004). OWL web ontology language semantics and abstract syntax. *W3C recommendation*, *10*.

Pauwels, P., De Meyer, R., & Van Campenhout, J. (2010). Interoperability for the design and construction industry through semantic web technology. In*Semantic Multimedia* (pp. 143-158). Springer Berlin Heidelberg.

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., & Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. *Automation in Construction*,*20*(5), 506-518.

Pazlar, T., & Turk, Z. (2008). Interoperability in practice: geometric data exchance using the IFC standard.

Prud'Hommeaux, E., & Seaborne, A. (2008). SPARQL query language for RDF. *W3C recommendation*, *15*.

Radulovic, F., Poveda-Villalón, M., Vila-Suero, D., Rodríguez-Doncel, V., García-Castro, R., & Gómez-Pérez, A. (2015). Guidelines for Linked Data generation and publication: An example in building energy consumption.*Automation in Construction*.

Radulovic, F., Poveda-Villalón, M., Vila-Suero, D., Rodríguez-Doncel, V., García-Castro, R., & Gómez-Pérez, A. (2015). Guidelines for Linked Data generation and publication: An example in building energy consumption. Automation in Construction, 57, 178-187.

Raskino, M., Fenn, J., & Linden, A. (2005). Extracting value from the massively connected world of 2015. *Gartner Res., Stamford, CT, USA, Tech. Rep. G*, *125949*.

Rees, J. A. (2012). Providing and Discovering URI Documentation. Editor's Draft 2 February 2012. *W3C. http://www. w3. org/2001/tag/awwsw/issue57/latest*.

Riaz, Z., Arslan, M., Kiani, A. K., & Azhar, S. (2014). CoSMoS: A BIM and wireless sensor based integrated solution for worker safety in confined spaces. *Automation in construction*, *45*, 96-106.

Russomanno, D. J., Kothari, C., & Thomas, O. (2005, March). Sensor ontologies: from shallow to deep models. In *System Theory, 2005. SSST'05. Proceedings of the Thirty-Seventh Southeastern Symposium on* (pp. 107-112). IEEE.

SEMIC (2012).10 rules for persistent URIs. Tech. Rep. Semantic Interoperability Community, European Commission.

Shannon, V. (2006). A'more revolutionary'Web. *International Herald Tribune*,*24*.

Sheth, A., Henson, C., & Sahoo, S. S. (2008). Semantic sensor web.*Internet Computing, IEEE*, *12*(4), 78-83.

Smart Appliances. (2013). Smart Appliances REFerence(SAREF) ontology. Retrieved from https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology. (Last retrieved on 21.06.2016).

Tech, I. A. HVAC Handbook–Thermal Comfort, 2002. *INNOVA Air Tech*.

Teicholz, P. (Ed.). (2013). *BIM for facility managers*. John Wiley & Sons.

Underwood, J., & Isikdag, U. (2011). Emerging technologies for BIM 2.0.*Construction Innovation*, *11*(3), 252-258.

Web, S. Resource Description Framework (RDF). *2004-02-10)[2011-04-07]. http://www. w3. org/RDF*.

Wei, W., & Barnaghi, P. (2009). Semantic annotation and reasoning for sensor data. In *Smart Sensing and Context* (pp. 66-76). Springer Berlin Heidelberg.

Woo, J. H., Diggelman, C., & Abushakra, B. (2011). BIM-based energy monitoring with XML parsing engine. *Proceeding of the 28th ISARC, Seoul, Korea*, 544-545.

World Wide Web Consortium. (2011). WGS84 Geo Positioning: an RDF vocabulary. *Retrieved on*, *21*.

Yan, H., & Damian, P. (2008, October). Benefits and barriers of building information modelling. In *12th International conference on computing in civil and building engineering* (Vol. 161).

Yin, H. (2010). Building management system to support building renovation.

Yurchyshyna, A., & Zarli, A. (2009). An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction. *Automation in Construction*, *18*(8), 1084-1098.

Appendix A Jupyter Notebook data download coding example

```
import numpy as np
import pandas.io.sql as sql
import psycopg2 as pg
import pandas as pd
import csv

# Lets use some data from the vertigo floor 6 project
# connect with database
con = pg.connect(host='archbpslindev2.bwk.tue.nl',
                 database='vertigo',
                 user='bpsstudent',
                 password='xxxxx')

sql_2015 = \
                            '''
                            SELECT dt, dt06 FROM floor6.comfort
                            WHERE date_part('year',dt) = 2015
                            AND ((date_part('hour', dt) = 9) AND
(date_part('minute', dt) = 30)
                                 OR (date_part('hour', dt) = 13) AND
(date_part('minute', dt) = 0)
                                 OR (date_part('hour', dt) = 17) AND
(date_part('minute', dt) = 30))
                            ORDER BY dt
                            '''
# read data from database in pandas dataframe
data= sql.read_sql(sql_2015, con, index_col=None)

print(data)

data.to_csv('/media/sf_ubuntu/DT06.csv')
```

Appendix B Self-defined RDF ontology schema (before data linking process)

```
@prefix sensor: <http://www.tue.nl/sensor#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix sensorM03: <http://Vertigo/floor6/sensorM03#> .
@prefix saref: <C:\Users\s146559\Desktop\Sensor Data\New\Sensor-
Description-csv.ttl> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix outsideTemperature: <http://Vertigo/OutsideTemperature#> .
@prefix Saref: <https://w3id.org/saref#> .
@prefix sensorM40: <http://Vertigo/floor6/sensorM40#> .
@prefix sensorM52: <http://Vertigo/floor6/sensorM52#> .
@prefix sensorM53: <http://Vertigo/floor6/sensorM53#> .
@prefix sensorM38: <http://Vertigo/floor6/sensorM38#> .
@prefix sensorM39: <http://Vertigo/floor6/sensorM39#> .
@prefix sensorDT06: <http://Vertigo/floor6/sensorDT06#> .
@prefix sensorDescription: <http://Vertigo/floor6/SensorDescription#> .
@prefix sensorDT07: <http://Vertigo/floor6/sensorDT07#> .
@prefix sensorDT08: <http://Vertigo/floor6/sensorDT08#> .
@prefix sensorDT03: <http://Vertigo/floor6/sensorDT03#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix inst:
<http://linkedbuildingdata.net/ifc/resources20160614_154955/> .
@prefix sensorM51: <http://Vertigo/floor6/sensorM51#> .

sensor:sensorRepresentation
    rdf:type owl:class
    rdfs:comment "sensor representation for each sensor"^^xsd:string .

sensor:sensorValue
    rdf:type owl:class;
    rdfs:comment "record sensing value of sensor DT03 at a specific
time"^^xsd:string ;
    sensor:isSensingValueOf sensor:sensorRepresentation;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

<http://Vertigo/floor6/sensorDT03#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT03 at a specific
time"^^xsd:string ;
    sensor:isSensingValueOf sensorDT03:DT03 ;
    Saref:hasSensingTime xsd:dateTime ;
    owl:hasValue xsd:double .

sensorDT03:DT03
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT03"^^xsd:string.


<http://Vertigo/floor6/sensorDT06#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT06 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorDT06:DT06;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorDT06:DT06
```

```
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT06"^^xsd:string.


<http://Vertigo/floor6/sensorDT07#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT07 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorDT07:DT07;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorDT07:DT07
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT07"^^xsd:string.


<http://Vertigo/floor6/sensorDT08#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT08 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorDT08:DT08;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorDT08:DT08
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT08"^^xsd:string.



<http://Vertigo/floor6/sensorM03#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M03 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM03:M03;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM03:M03
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M03"^^xsd:string.


<http://Vertigo/floor6/sensorM38#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M38 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM38:M38;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM38:M38
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M38"^^xsd:string.


<http://Vertigo/floor6/sensorM39#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M39 at a specific
time"^^xsd:string;
```

```
    sensor:isSensingValueOf sensorM39:M39;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM39:M39
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M39"^^xsd:string.


<http://Vertigo/floor6/sensorM40#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M40 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM40:M40;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM40:M40
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M40"^^xsd:string.


<http://Vertigo/floor6/sensorM51#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M51 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM51:M51;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM51:M51
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M51"^^xsd:string.


<http://Vertigo/floor6/sensorM52#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M52 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM52:M52;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM52:M52
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M52"^^xsd:string.


<http://Vertigo/floor6/sensorM53#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M53 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM53:M53;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM53:M53
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M53"^^xsd:string.
```

```
<http://Vertigo/OutsideTemperature#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of outside temperature sensor at a
specific time"^^xsd:string;
    sensor:isSensingValueOf outsideTemperature:outsideTemperature;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

outsideTemperature:outsideTemperature
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of outside temperature
sensor"^^xsd:string.

sensorDescription:sensorDescription
    rdf:type owl:Class ;
    rdfs:comment "sensor description for sensors "^^xsd:string ;
    sensorDescription:introduces  sensor:sensorRepresentation ;
    Saref:hasName sensorDescription:sensorName ;
    Saref:hasSingularUnit xsd:string ;
    Saref:hasSensorType xsd:string ;
    geo:location sensorDescription:room;
    sensorDescription:locationHeight xsd:string ;
    Saref:hasCategory sensorDescription:category.

<http://Vertigo/floor6/SensorDescription#0>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor DT03"^^xsd:string ;
    sensorDescription:introduces  sensorDT03:DT03 ;
    Saref:hasName sensorDescription:DT03 ;
    Saref:hasSingularUnit "m/s"^^xsd:string ;
    Saref:hasSensorType "Air speed at 1.1 m"^^xsd:string ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "1.1m"^^xsd:string ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#1>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor DT06"^^xsd:string ;
    sensorDescription:introduces  sensorDT06:DT06 ;
    Saref:hasName sensorDescription:DT06 ;
    Saref:hasSingularUnit ""?C"^^xsd:string ;
    sensorDescription:hasSensingRange "-30.0 - 65.0" ;
    Saref:hasSensorType "Air temperature at 1.1 m"^^xsd:string ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "1.1m"^^xsd:string ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#2>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor DT07"^^xsd:string ;
    sensorDescription:introduces  sensorDT07:DT07 ;
    Saref:hasName sensorDescription:DT07 ;
    Saref:hasSingularUnit "?C" ;
    Saref:hasSensorType "Temperature black globe at 0.6 m" ;
    sensorDescription:hasSensingRange "-30.0 - 65.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "0.6m" ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#3>
    rdf:type sensorDescription:sensorDescription ;
```

```
    rdfs:comment "a sensor description for sensor DT08"^^xsd:string ;
    sensorDescription:introduces  sensorDT08:DT08 ;
    Saref:hasName sensorDescription:DT08 ;
    Saref:hasSingularUnit "%" ;
    Saref:hasSensorType "Relative Humidity at 0.6 m" ;
    sensorDescription:hasSensingRange "0.0 - 100.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "0.6m" ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#4>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M03"^^xsd:string ;
    sensorDescription:introduces  sensorM03:M03 ;
    Saref:hasName sensorDescription:M03 ;
    Saref:hasSingularUnit "ppm" ;
    Saref:hasSensorType "CO2" ;
    sensorDescription:hasSensingRange "0 - 5000" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#5>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M38"^^xsd:string ;
    sensorDescription:introduces  sensorM38:M38 ;
    Saref:hasName sensorDescription:M38 ;
    Saref:hasSingularUnit "?C" ;
    Saref:hasSensorType "Temperature" ;
    sensorDescription:hasSensingRange "5 - 45" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.18> ;
    sensorDescription:locationHeight "0.75m" ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#6>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M39"^^xsd:string ;
    sensorDescription:introduces  sensorM39:M39 ;
    Saref:hasName sensorDescription:M39 ;
    Saref:hasSingularUnit "%" ;
    Saref:hasSensorType "Relatieve humidity" ;
    sensorDescription:hasSensingRange "0.0 - 100.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.18> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#7>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M40"^^xsd:string ;
    sensorDescription:introduces  sensorM40:M40 ;
    Saref:hasName sensorDescription:M40 ;
    Saref:hasSingularUnit "ppm" ;
    Saref:hasSensorType "CO2" ;
    sensorDescription:hasSensingRange "0 - 5000" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.18> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#8>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M51"^^xsd:string ;
    sensorDescription:introduces  sensorM51:M51 ;
    Saref:hasName sensorDescription:M51 ;
    Saref:hasSingularUnit "?C" ;
```

```
    Saref:hasSensorType "Temperature" ;
    sensorDescription:hasSensingRange "-30.0 - 65.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.20> ;
    sensorDescription:locationHeight "0.75m" ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#9>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M52"^^xsd:string ;
    sensorDescription:introduces  sensorM52:M52 ;
    Saref:hasName sensorDescription:M52 ;
    Saref:hasSingularUnit "%" ;
    Saref:hasSensorType "Relatieve humidity" ;
    sensorDescription:hasSensingRange "0.0 - 100.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.20> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#10>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M53"^^xsd:string ;
    sensorDescription:introduces  sensorM53:M53 ;
    Saref:hasName sensorDescription:M53 ;
    Saref:hasSingularUnit "ppm" ;
    Saref:hasSensorType "CO2" ;
    sensorDescription:hasSensingRange "0 - 5000" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.20> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#11>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for outside temperature
sensor"^^xsd:string ;
    sensorDescription:introduces  outsideTemperature:outsideTemperature ;
    Saref:hasName sensorDescription:T_amb_avg ;
    Saref:hasSingularUnit "?C" ;
    Saref:hasSensorType "Outside temperature" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Vertigo+roof> ;
    Saref:hasCategory sensorDescription:DL_SMS_Mete .

sensorDescription:sensorName
    rdf:type owl:Class ;
    rdfs:comment "the description URIs for sensors"^^xsd:string .

sensorDescription:DT03
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT03"^^xsd:string .

sensorDescription:DT06
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT06"^^xsd:string .

sensorDescription:DT07
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT07"^^xsd:string .

sensorDescription:DT08
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT0"^^xsd:string .

sensorDescription:M03
    rdf:type sensorDescription:sensorName ;
```

```
    rdfs:comment "the description URI for sensor M03"^^xsd:string .

sensorDescription:M38
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M38"^^xsd:string .

sensorDescription:M39
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M39"^^xsd:string .

sensorDescription:M40
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M40"^^xsd:string .

sensorDescription:M51
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M51"^^xsd:string .

sensorDescription:M52
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M52"^^xsd:string .

sensorDescription:M53
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M53"^^xsd:string .

sensorDescription:T_amb_avg
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor T_amb_avg"^^xsd:string .

sensorDescription:room
    rdf:type owl:Class;
    rdfs:comment "the room URI where the sensor is located"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Room+6.08>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for room 6.08"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Room+6.18>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for room 6.18"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Room+6.20>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for room 6.20"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Vertigo+roof>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for Vertigo roof"^^xsd:string .

sensorDescription:category
    rdf:type owl:Class ;
    rdfs:comment "the category for different sensors"^^xsd:string .

sensorDescription:ComfortStatief
    rdf:type sensorDescription:category ;
    rdfs:comments "the log schedule is ComfortStatief"^^xsd:string .

sensorDescription:Modbus
    rdf:type sensorDescription:category ;
    rdfs:comments "the log schedule is Modbus"^^xsd:string .
```

```
sensorDescription:DL_SMS_Mete
    rdf:type sensorDescription:category ;
    rdfs:comments "the log schedule is DL_SMS_Mete"^^xsd:string .

sensor:isSensingValueOf
    rdf:type owl:ObjectProperty ;
    rdfs:comment "connection between sensor values and sensor
representation"^^xsd:string ;
    rdfs:domain sensor:sensorValue ;
    rdfs:range sensor:sensorRepresentation .
```

Appendix C Apache Jena coding

```
import org.apache.jena.rdf.model.*;
import org.apache.jena.util.FileManager;
import org.apache.jena.query.*;

import java.io.*;
import org.apache.log4j.BasicConfigurator;

public class LinkedData {

    static final String inputFilePath1 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\DT03-csv.ttl";
    static final String inputFilePath2 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\DT06-csv.ttl";
    static final String inputFilePath3 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\DT07-csv.ttl";
    static final String inputFilePath4 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\DT08-csv.ttl";
    static final String inputFilePath5 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M03-csv.ttl";
    static final String inputFilePath6 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M38-csv.ttl";
    static final String inputFilePath7 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M39-csv.ttl";
    static final String inputFilePath8 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M40-csv.ttl";
    static final String inputFilePath9 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M51-csv.ttl";
    static final String inputFilePath10 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M52-csv.ttl";
    static final String inputFilePath11 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\M53-csv.ttl";
    static final String inputFilePath12 = "C:\\Users\\s146559\\Desktop\\New
folder\\Vertigo floor6 building model.ttl";
    static final String inputFilePath13 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\Outside-Temperature-
csv.ttl";
    static final String inputFilePath14 =
"C:\\Users\\s146559\\Desktop\\Sensor Data\\New\\Sensor-Description-
csv.ttl";

    public static void main(String args[]) throws FileNotFoundException{

       // args: contains the date to be queried

       //create 14 models
        Model model1 =
ModelFactory.createDefaultModel().read(inputFilePath1);
        Model model2 =
ModelFactory.createDefaultModel().read(inputFilePath2);
        Model model3 =
ModelFactory.createDefaultModel().read(inputFilePath3);
        Model model4 =
ModelFactory.createDefaultModel().read(inputFilePath4);
        Model model5 =
ModelFactory.createDefaultModel().read(inputFilePath5);
        Model model6 =
ModelFactory.createDefaultModel().read(inputFilePath6);
```

```
        Model model7 =
ModelFactory.createDefaultModel().read(inputFilePath7);
        Model model8 =
ModelFactory.createDefaultModel().read(inputFilePath8);
        Model model9 =
ModelFactory.createDefaultModel().read(inputFilePath9);
        Model model10 =
ModelFactory.createDefaultModel().read(inputFilePath10);
        Model model11 =
ModelFactory.createDefaultModel().read(inputFilePath11);
        Model model12 =
ModelFactory.createDefaultModel().read(inputFilePath12);
        Model model13 =
ModelFactory.createDefaultModel().read(inputFilePath13);
        Model model14 =
ModelFactory.createDefaultModel().read(inputFilePath14);


        //create a new model and write all the triples into the new one
        Model blob = ModelFactory.createDefaultModel();
        Model[] models = {model1, model2, model3, model4, model5, model6,
        model7, model8, model9, model10, model11, model12, model13,
model14};

        for(Model part: models){
            blob.add(part );
        }


        //build links between resources
        Model linking = ModelFactory.createDefaultModel();
        Model model = createLinks(linking, blob);

        //write the model into a file
        OutputStream out = new FileOutputStream(new
File("C:\\Users\\s146559\\Desktop\\New folder\\model.ttl"));
        model.write(out, "TURTLE");

        //SPARQL for human thermal comfort monitoring

        String[] airSpeedDate = new String[400];
        double[] airSpeedValue = new double[400];
        String[] temperatureDate = new String[400];
        double[] temperatureValue = new double[400];
        String[] blackGlobeDate = new String[400];
        double[] blackGlobeValue = new double[400];
        String[] humidityDate = new String[400];
        double[] humidityValue = new double[400];

        queryRoom608ThermalComfort(model, airSpeedDate, temperatureDate,
blackGlobeDate, humidityDate, airSpeedValue,
                temperatureValue, blackGlobeValue, humidityValue);


        //calculate for human thermal comfort
        double[] PMVRoom608 = new double[400];
        double[] PPDRoom608 = new double[400];

        for(int i = 0; i < airSpeedDate.length; i++){
            double m = 70;
            double w = 0;
```

```
            double pw = 0;
            if(temperatureValue[i] >= 18 && temperatureValue[i] < 21){
                pw = (temperatureValue[i] - 18) / 3 * 4 + 18;
            }
            else if(temperatureValue[i] >= 21 && temperatureValue[i] < 24){
                pw = (temperatureValue[i] - 21) / 3 * 4.6 + 21;
            }
            else if(temperatureValue[i] >= 24 && temperatureValue[i] < 27){
                pw = (temperatureValue[i] - 24) / 3 * 5.4 + 24;
            }
            else if(temperatureValue[i] >= 27 && temperatureValue[i] < 29){
                pw = (temperatureValue[i] - 27) / 3 * 6 + 27;
            }
            double pa = humidityValue[i] / 1000 * pw;
            double hc = 12.1 * Math.pow(airSpeedValue[i], 0.5);
            double icl = 1.0;
            double rcl = 0.155 * icl;
            double fcl = (1.0 + 0.2 * icl) / (1.05 + 0.1 *icl);
            double tcl = 35.7 - 0.0275 * (m - w) - rcl * ((m - w) - 3.05 *
(5.73 - 0.007 *(m - w) - pa) -
            0.42 * ((m - w) - 58.15) - 0.0173 * m * (5.87 - pa) - 0.0014 *
m * (34 - temperatureValue[i]));
            double tr = Math.pow(Math.pow((blackGlobeValue[i] + 273), 4) +
2.5 * Math.pow(10, 8) * Math.pow(airSpeedValue[i], 0.6) *
                    (blackGlobeValue[i] - temperatureValue[i]), 0.25) -273;

            //System.out.println("tcl: " + tcl + " tr: " + tr + " fcl: " +
fcl + " rcl: " + rcl + " hc: " + hc + " pa: " + pa);
            //System.out.println("blackGlobe: " + blackGlobeValue[i] + "
airSpeedValue: " + airSpeedValue[i] + " temperatureValue: " +
temperatureValue[i] + " humidity: " + humidityValue[i]);
            //System.out.println("Date: " + airSpeedDate[i] + ", Room6.08
has thermal comfort: " + PMVRoom608[i] +
            //" , and the Predicted Percentage of Dissatisfied is " +
PPDRoom608[i]);
            PMVRoom608[i] = (0.303 * Math.exp(-0.036 * m) + 0.028) * ((m -
w) - 3.96e-8 * fcl * (Math.pow((tcl + 273), 4) -
                    Math.pow((tr + 273), 4)) - fcl * hc * (tcl -
temperatureValue[i]) - 3.05 * (5.73 - 0.007 * (m - w) - pa) - 0.42 * ((m -
w) - 58.15)
                    -0.0173 * m * (5.87 - pa) - 0.0014 * m * (34 -
temperatureValue[i]));
            PPDRoom608[i] = 100 - 95 * Math.exp(-(0.3353 *
Math.pow(PMVRoom608[i], 4) + 0.2179 * Math.pow(PMVRoom608[i], 2)));
            System.out.println("PMV- " + PMVRoom608[i] + "  PPD- " +
PPDRoom608[i]);

        }

        //PMV and PPD for feb, April, July, October at 13:00
        double[] results = PMVPPD(airSpeedDate, PMVRoom608, 28, "2015-02",
"13:00:00");
        double PMVFeb = results[0];
        double PPDFeb = results[1];

        double[] results1 = PMVPPD(airSpeedDate, PMVRoom608, 30, "2015-04",
"13:00:00");
        double PMVApril = results1[0];
        double PPDApril = results1[1];
```

```
        double[] results2 = PMVPPD(airSpeedDate, PMVRoom608, 31, "2015-07",
"13:00:00");
        double PMVJuly = results2[0];
        double PPDJuly = results2[1];

        double[] results3 = PMVPPD(airSpeedDate, PMVRoom608, 31, "2015-10",
"13:00:00");
        double PMVOct = results3[0];
        double PPDOct = results3[1];


        System.out.println("PMVFeb, PPDFeb: " + PMVFeb + " " + PPDFeb +
"PMVApril, PPDApril: " +
                PMVApril + " " + PPDApril + "PMVJuly, PPDJuly: " + PMVJuly
+ " " + PPDJuly +
                "PMVOct, PPDOct: " + PMVOct + " " + PPDOct);

        //PMV and PPD at 09:30, 13:00, 17:30 at April
        double[] results4 = PMVPPD(airSpeedDate, PMVRoom608, 30, "2015-04",
"09:30:00");
        double PMV0930 = results4[0];
        double PPD0930 = results4[1];

        double[] results5 = PMVPPD(airSpeedDate, PMVRoom608, 30, "2015-04",
"13:00:00");
        double PMV1300 = results5[0];
        double PPD1300 = results5[1];

        double[] results6 = PMVPPD(airSpeedDate, PMVRoom608, 30, "2015-04",
"17:30:00");
        double PMV1730 = results6[0];
        double PPD1730 = results6[1];


        System.out.println("PMV0930, PPD0930: " + PMV0930 + " " + PPD0930 +
"PMV1300, PPD1300: " +
                PMV1300 + " " + PPD1300 + "PMV1730, PPD1730: " + PMV1730 +
" " + PPD1730);



    }

    public static Model createLinks(Model linking, Model blob){

        //DT03 with description
        Resource DT03 =
linking.createResource("http://Vertigo/floor6/sensorDT03#DT03");
        Property introduce =
linking.createProperty("http://Vertigo/floor6/SensorDescription#",
"introduces");
        Resource DT03Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#0");
        DT03Description.addProperty(introduce, DT03);

        //DT06 with description
        Resource DT06 =
linking.createResource("http://Vertigo/floor6/sensorDT06#DT06");
        Resource DT06Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#1");
        DT06Description.addProperty(introduce, DT06);
```

```
        //DT07 with description
        Resource DT07 =
linking.createResource("http://Vertigo/floor6/sensorDT07#DT07");
        Resource DT07Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#2");
        DT07Description.addProperty(introduce, DT07);

        //DT08 with description
        Resource DT08 =
linking.createResource("http://Vertigo/floor6/sensorDT08#DT08");
        Resource DT08Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#3");
        DT08Description.addProperty(introduce, DT08);

        //M03 with description
        Resource M03 =
linking.createResource("http://Vertigo/floor6/sensorM03#M03");
        Resource M03Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#4");
        M03Description.addProperty(introduce, M03);

        //M38 with description
        Resource M38 =
linking.createResource("http://Vertigo/floor6/sensorM38#M38");
        Resource M38Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#5");
        M38Description.addProperty(introduce, M38);

        //M39 with description
        Resource M39 =
linking.createResource("http://Vertigo/floor6/sensorM39#M39");
        Resource M39Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#6");
        M39Description.addProperty(introduce, M39);

        //M40 with description
        Resource M40 =
linking.createResource("http://Vertigo/floor6/sensorM40#M40");
        Resource M40Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#7");
        M40Description.addProperty(introduce, M40);

        //M51 with description
        Resource M51 =
linking.createResource("http://Vertigo/floor6/sensorM51#M51");
        Resource M51Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#8");
        M51Description.addProperty(introduce, M51);

        //M52 with description
        Resource M52 =
linking.createResource("http://Vertigo/floor6/sensorM52#M52");
        Resource M52Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#9");
        M52Description.addProperty(introduce, M52);

        //M53 with description
        Resource M53 =
linking.createResource("http://Vertigo/floor6/sensorM53#M53");
```

```
        Resource M53Description =
linking.createResource("http://Vertigo/floor6/SensorDescription#10");
        M53Description.addProperty(introduce, M53);

        //T_amb_avg with description
        Resource T_amb_avg =
linking.createResource("http://Vertigo/OutsideTemperature#outsideTemperatur
e");
        Resource T_amb_avgDescription =
linking.createResource("http://Vertigo/floor6/SensorDescription#11");
        T_amb_avgDescription.addProperty(introduce, T_amb_avg);

        //IFC room 6.08 with sensor description
        Resource room608 =
linking.createResource("http://Vertigo/floor6/SensorDescription#Room+6.08")
;
        Property hasIfcLabel =
linking.createProperty("http://Vertigo/floor6/SensorDescription#",
"hasIfcLabel");
        Resource Label608 =
linking.createResource("http://linkedbuildingdata.net/ifc/resources20160530
_160210#IfcLabel_14517");
        room608.addProperty(hasIfcLabel, Label608);

        //IFC room 6.18 with sensor description
        Resource room618 =
linking.createResource("http://Vertigo/floor6/SensorDescription#Room+6.18")
;
        Resource Label618 =
linking.createResource("http://linkedbuildingdata.net/ifc/resources20160530
_160210#IfcLabel_14002");
        room618.addProperty(hasIfcLabel, Label618);

        //IFC room 6.20 with sensor description
        Resource room620 =
linking.createResource("http://Vertigo/floor6/SensorDescription#Room+6.20")
;
        Resource Label620 =
linking.createResource("http://linkedbuildingdata.net/ifc/resources20160530
_160210#IfcLabel_14261");
        room620.addProperty(hasIfcLabel, Label620);


        Model model = blob.union(linking);

        return model;

    }

    public static void queryRoom608ThermalComfort(Model model, String[]
airSpeedDate, String[] temperatureDate, String[] blackGlobeDate,
            String[] humidityDate, double[] airSpeedValue, double[]
temperatureValue, double[] blackGlobeValue, double[] humidityValue){

        //query for air speed(DT03) at 02/2015, 04/2015, 07/2015, 10/2015
        String queryDT03 = "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
```

```
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "SELECT ?date ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Air speed at 1.1 m\". " +
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcLabel  express:hasString   \"Room 6.08\" . " +
                "  ?c  a  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime ?date . " +
                "  FILTER (((?date >= \"2015-02-
01T09:30:00+01:00\"^^xsd:dateTime) && (?date < \"2015-03-
01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + " ((?date >= \"2015-04-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-05-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-07-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-08-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-10-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-11-01T09:30:00+01:00\"^^xsd:dateTime))) " +
                "}\n" +
                "ORDER BY ?date";
        Query qDT03 = QueryFactory.create(queryDT03);
        QueryExecution qExeDT03 = QueryExecutionFactory.create(qDT03,
model);
        try{
            ResultSet resultDT03 = qExeDT03.execSelect();
            int i = 0;
                while(resultDT03.hasNext()){
                    QuerySolution soln = resultDT03.nextSolution();
                    Literal date = soln.getLiteral("date");
                    Literal value = soln.getLiteral("value");
                    System.out.println("airSpeedRoom608- " + "Date: " +
date + "   Value: " + value);
                    String[] dateParts = date.getString().split("^^");
                    airSpeedDate[i] = dateParts[0];
                    String[] valueParts = value.getString().split("^^");
                    double temp = Double.parseDouble(valueParts[0]);
                    airSpeedValue[i] = temp;
                    i++;
            }
                for(int j = 0; j < i; j++){
                    System.out.println("airSpeedRoom608-  " +
airSpeedDate[j] + "  " + airSpeedValue[j]);
                }
        }finally{
            qExeDT03.close();
        }

        //query for temperature(DT06) on 01/02/2015, 01/04/2015,
01/07/2015, 01/10/2015
        String queryDT06 = "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
```

```
                "SELECT ?date ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Air temperature at 1.1 m\". "
+
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcLabel  express:hasString   \"Room 6.08\" . " +
                "  ?c  a  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime ?date . " +
                "  FILTER (((?date >= \"2015-02-
01T09:30:00+01:00\"^^xsd:dateTime) && (?date < \"2015-03-
01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + " ((?date >= \"2015-04-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-05-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-07-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-08-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-10-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-11-01T09:30:00+01:00\"^^xsd:dateTime))) " +
                "}\n" +
                "ORDER BY ?date";
        Query qDT06 = QueryFactory.create(queryDT06);
        QueryExecution qExeDT06 = QueryExecutionFactory.create(qDT06,
model);
        try{
            ResultSet resultDT06 = qExeDT06.execSelect();
            int i = 0;
                while(resultDT06.hasNext()){
                    QuerySolution soln = resultDT06.nextSolution();
                    Literal date = soln.getLiteral("date");
                    Literal value = soln.getLiteral("value");
                    System.out.println("temperatureRoom608-  " + "Date: " +
date + "   Value: " + value);
                    String[] dateParts = date.getString().split("^^");
                    temperatureDate[i] = dateParts[0];
                    String[] valueParts = value.getString().split("^^");
                    double temp = Double.parseDouble(valueParts[0]);
                    temperatureValue[i] = temp;
                    i++;
            }
                for(int j = 0; j < i; j++){
                    System.out.println("temperatureRoom608-  " +
temperatureDate[j] + "  " + temperatureValue[j]);
                }
        }finally{
            qExeDT06.close();
        }

        //query for temperature black globe(DT07) on 01/02/2015,
01/04/2015, 01/07/2015, 01/10/2015
        String queryDT07 = "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "SELECT ?date ?value " +
                "WHERE " +
```

```
                "{ ?a  Saref:hasSensorType  \"Temperature black globe at
0.6 m\" . " +
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcLabel  express:hasString   \"Room 6.08\" . " +
                "  ?c  a  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime ?date . " +
                "  FILTER (((?date >= \"2015-02-
01T09:30:00+01:00\"^^xsd:dateTime) && (?date < \"2015-03-
01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + " ((?date >= \"2015-04-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-05-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-07-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-08-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-10-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-11-01T09:30:00+01:00\"^^xsd:dateTime))) " +
                "}\n" +
                "ORDER BY ?date";
        Query qDT07 = QueryFactory.create(queryDT07);
        QueryExecution qExeDT07 = QueryExecutionFactory.create(qDT07,
model);
        try{
            ResultSet resultDT07 = qExeDT07.execSelect();
            int i = 0;
                while(resultDT07.hasNext()){
                    QuerySolution soln = resultDT07.nextSolution();
                    Literal date = soln.getLiteral("date");
                    Literal value = soln.getLiteral("value");
                    System.out.println("blackGlobeRoom608-  " + "Date: " +
date + "   Value: " + value);
                    String[] dateParts = date.getString().split("^^");
                    blackGlobeDate[i] = dateParts[0];
                    String[] valueParts = value.getString().split("^^");
                    double temp = Double.parseDouble(valueParts[0]);
                    blackGlobeValue[i] = temp;
                    i++;
            }
                for(int j = 0; j < i; j++){
                    System.out.println("blackGlobeRoom608-  " +
blackGlobeDate[j] + "  " + blackGlobeValue[j]);
                }
        }finally{
            qExeDT07.close();
        }

        //query for humidity(DT08) on 01/02/2015, 01/04/2015, 01/07/2015,
01/10/2015
        String queryDT08 = "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "SELECT ?date ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Relative Humidity at 0.6
m\" . " +
```

```
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcLabel  express:hasString   \"Room 6.08\" . " +
                "  ?c  a  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime ?date . " +
                "  FILTER (((?date >= \"2015-02-
01T09:30:00+01:00\"^^xsd:dateTime) && (?date < \"2015-03-
01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + " ((?date >= \"2015-04-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-05-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-07-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-08-01T09:30:00+01:00\"^^xsd:dateTime)) ||"
                + "((?date >= \"2015-10-01T09:30:00+01:00\"^^xsd:dateTime)
&& (?date < \"2015-11-01T09:30:00+01:00\"^^xsd:dateTime))) " +
                "}\n" +
                "ORDER BY ?date";
        Query qDT08 = QueryFactory.create(queryDT08);
        QueryExecution qExeDT08 = QueryExecutionFactory.create(qDT08,
model);
        try{
            ResultSet resultDT08 = qExeDT08.execSelect();
            int i = 0;
                while(resultDT08.hasNext()){
                    QuerySolution soln = resultDT08.nextSolution();
                    Literal date = soln.getLiteral("date");
                    Literal value = soln.getLiteral("value");
                    System.out.println("humidityRoom608-  " + "Date: " +
date + "   Value: " + value);
                    String[] dateParts = date.getString().split("^^");
                    humidityDate[i] = dateParts[0];
                    String[] valueParts = value.getString().split("^^");
                    double temp = Double.parseDouble(valueParts[0]);
                    humidityValue[i] = temp;
                    i++;
            }
                for(int j = 0; j < i; j++){
                    System.out.println("humidityRoom608-  " +
humidityDate[j] + "  " + humidityValue[j]);
                }
        }finally{
            qExeDT08.close();
        }

    }



    public static double[] PMVPPD(String[] airSpeedDate, double[]
PMVRoom608, int date, String day, String time){
        double PMVSum = 0;
        for(int i = 0; i < airSpeedDate.length; i++){
            if (airSpeedDate[i] == null) break;
            if(airSpeedDate[i].contains(day)){
                if(airSpeedDate[i].contains(time)){
                    PMVSum += PMVRoom608[i];
                }
            }
        }
        double[] PMVPPD = new double[2];
```

```
        PMVPPD[0] = PMVSum/date;
        PMVPPD[1] = 100 - 95 * Math.exp(-(0.3353 * Math.pow(PMVPPD[0], 4) +
0.2179 * Math.pow(PMVPPD[1], 2)));
        return PMVPPD;
        }

}
```

Appendix D Self-defined RDF ontology schema (after data linking process)

```
@prefix sensor: <http://www.tue.nl/sensor#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix sensorM03: <http://Vertigo/floor6/sensorM03#> .
@prefix saref: <C:\Users\s146559\Desktop\Sensor Data\New\Sensor-
Description-csv.ttl> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix outsideTemperature: <http://Vertigo/OutsideTemperature#> .
@prefix Saref: <https://w3id.org/saref#> .
@prefix sensorM40: <http://Vertigo/floor6/sensorM40#> .
@prefix sensorM52: <http://Vertigo/floor6/sensorM52#> .
@prefix sensorM53: <http://Vertigo/floor6/sensorM53#> .
@prefix sensorM38: <http://Vertigo/floor6/sensorM38#> .
@prefix sensorM39: <http://Vertigo/floor6/sensorM39#> .
@prefix sensorDT06: <http://Vertigo/floor6/sensorDT06#> .
@prefix sensorDescription: <http://Vertigo/floor6/SensorDescription#> .
@prefix sensorDT07: <http://Vertigo/floor6/sensorDT07#> .
@prefix sensorDT08: <http://Vertigo/floor6/sensorDT08#> .
@prefix sensorDT03: <http://Vertigo/floor6/sensorDT03#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix inst:
<http://linkedbuildingdata.net/ifc/resources20160614_154955/> .
@prefix sensorM51: <http://Vertigo/floor6/sensorM51#> .

sensor:sensorRepresentation
    rdf:type owl:class
    rdfs:comment "sensor representation for each sensor"^^xsd:string .

sensor:sensorValue
    rdf:type owl:class;
    rdfs:comment "record sensing value of sensor DT03 at a specific
time"^^xsd:string ;
    sensor:isSensingValueOf sensor:sensorRepresentation;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

<http://Vertigo/floor6/sensorDT03#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT03 at a specific
time"^^xsd:string ;
    sensor:isSensingValueOf sensorDT03:DT03 ;
    Saref:hasSensingTime xsd:dateTime ;
    owl:hasValue xsd:double .

sensorDT03:DT03
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT03"^^xsd:string.


<http://Vertigo/floor6/sensorDT06#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT06 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorDT06:DT06;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorDT06:DT06
```

```
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT06"^^xsd:string.


<http://Vertigo/floor6/sensorDT07#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT07 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorDT07:DT07;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorDT07:DT07
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT07"^^xsd:string.


<http://Vertigo/floor6/sensorDT08#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor DT08 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorDT08:DT08;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorDT08:DT08
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of DT08"^^xsd:string.



<http://Vertigo/floor6/sensorM03#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M03 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM03:M03;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM03:M03
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M03"^^xsd:string.


<http://Vertigo/floor6/sensorM38#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M38 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM38:M38;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM38:M38
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M38"^^xsd:string.


<http://Vertigo/floor6/sensorM39#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M39 at a specific
time"^^xsd:string;
```

```
    sensor:isSensingValueOf sensorM39:M39;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM39:M39
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M39"^^xsd:string.


<http://Vertigo/floor6/sensorM40#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M40 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM40:M40;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM40:M40
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M40"^^xsd:string.


<http://Vertigo/floor6/sensorM51#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M51 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM51:M51;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM51:M51
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M51"^^xsd:string.


<http://Vertigo/floor6/sensorM52#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M52 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM52:M52;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM52:M52
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M52"^^xsd:string.


<http://Vertigo/floor6/sensorM53#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of sensor M53 at a specific
time"^^xsd:string;
    sensor:isSensingValueOf sensorM53:M53;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

sensorM53:M53
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of M53"^^xsd:string.
```

```
<http://Vertigo/OutsideTemperature#0>
    rdf:type sensor:sensorValue ;
    rdfs:comment "a sensing value of outside temperature sensor at a
specific time"^^xsd:string;
    sensor:isSensingValueOf outsideTemperature:outsideTemperature;
    Saref:hasSensingTime xsd:dateTime;
    owl:hasValue xsd:double.

outsideTemperature:outsideTemperature
    rdf:type sensor:sensorRepresentation ;
    rdfs:comment "the representation of outside temperature
sensor"^^xsd:string.

sensorDescription:sensorDescription
    rdf:type owl:Class ;
    rdfs:comment "sensor description for sensors "^^xsd:string ;
    sensorDescription:introduces  sensor:sensorRepresentation ;
    Saref:hasName sensorDescription:sensorName ;
    Saref:hasSingularUnit xsd:string ;
    Saref:hasSensorType xsd:string ;
    geo:location sensorDescription:room;
    sensorDescription:locationHeight xsd:string ;
    Saref:hasCategory sensorDescription:category.

<http://Vertigo/floor6/SensorDescription#0>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor DT03"^^xsd:string ;
    sensorDescription:introduces  sensorDT03:DT03 ;
    Saref:hasName sensorDescription:DT03 ;
    Saref:hasSingularUnit "m/s"^^xsd:string ;
    Saref:hasSensorType "Air speed at 1.1 m"^^xsd:string ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "1.1m"^^xsd:string ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#1>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor DT06"^^xsd:string ;
    sensorDescription:introduces  sensorDT06:DT06 ;
    Saref:hasName sensorDescription:DT06 ;
    Saref:hasSingularUnit ""?C"^^xsd:string ;
    sensorDescription:hasSensingRange "-30.0 - 65.0" ;
    Saref:hasSensorType "Air temperature at 1.1 m"^^xsd:string ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "1.1m"^^xsd:string ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#2>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor DT07"^^xsd:string ;
    sensorDescription:introduces  sensorDT07:DT07 ;
    Saref:hasName sensorDescription:DT07 ;
    Saref:hasSingularUnit "?C" ;
    Saref:hasSensorType "Temperature black globe at 0.6 m" ;
    sensorDescription:hasSensingRange "-30.0 - 65.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "0.6m" ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#3>
    rdf:type sensorDescription:sensorDescription ;
```

```
    rdfs:comment "a sensor description for sensor DT08"^^xsd:string ;
    sensorDescription:introduces  sensorDT08:DT08 ;
    Saref:hasName sensorDescription:DT08 ;
    Saref:hasSingularUnit "%" ;
    Saref:hasSensorType "Relative Humidity at 0.6 m" ;
    sensorDescription:hasSensingRange "0.0 - 100.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    sensorDescription:locationHeight "0.6m" ;
    Saref:hasCategory sensorDescription:ComfortStatief .

<http://Vertigo/floor6/SensorDescription#4>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M03"^^xsd:string ;
    sensorDescription:introduces  sensorM03:M03 ;
    Saref:hasName sensorDescription:M03 ;
    Saref:hasSingularUnit "ppm" ;
    Saref:hasSensorType "CO2" ;
    sensorDescription:hasSensingRange "0 - 5000" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.08> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#5>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M38"^^xsd:string ;
    sensorDescription:introduces  sensorM38:M38 ;
    Saref:hasName sensorDescription:M38 ;
    Saref:hasSingularUnit "?C" ;
    Saref:hasSensorType "Temperature" ;
    sensorDescription:hasSensingRange "5 - 45" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.18> ;
    sensorDescription:locationHeight "0.75m" ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#6>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M39"^^xsd:string ;
    sensorDescription:introduces  sensorM39:M39 ;
    Saref:hasName sensorDescription:M39 ;
    Saref:hasSingularUnit "%" ;
    Saref:hasSensorType "Relatieve humidity" ;
    sensorDescription:hasSensingRange "0.0 - 100.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.18> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#7>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M40"^^xsd:string ;
    sensorDescription:introduces  sensorM40:M40 ;
    Saref:hasName sensorDescription:M40 ;
    Saref:hasSingularUnit "ppm" ;
    Saref:hasSensorType "CO2" ;
    sensorDescription:hasSensingRange "0 - 5000" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.18> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#8>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M51"^^xsd:string ;
    sensorDescription:introduces  sensorM51:M51 ;
    Saref:hasName sensorDescription:M51 ;
    Saref:hasSingularUnit "?C" ;
```

```
    Saref:hasSensorType "Temperature" ;
    sensorDescription:hasSensingRange "-30.0 - 65.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.20> ;
    sensorDescription:locationHeight "0.75m" ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#9>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M52"^^xsd:string ;
    sensorDescription:introduces  sensorM52:M52 ;
    Saref:hasName sensorDescription:M52 ;
    Saref:hasSingularUnit "%" ;
    Saref:hasSensorType "Relatieve humidity" ;
    sensorDescription:hasSensingRange "0.0 - 100.0" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.20> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#10>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for sensor M53"^^xsd:string ;
    sensorDescription:introduces  sensorM53:M53 ;
    Saref:hasName sensorDescription:M53 ;
    Saref:hasSingularUnit "ppm" ;
    Saref:hasSensorType "CO2" ;
    sensorDescription:hasSensingRange "0 - 5000" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Room+6.20> ;
    Saref:hasCategory sensorDescription:Modbus .

<http://Vertigo/floor6/SensorDescription#11>
    rdf:type sensorDescription:sensorDescription ;
    rdfs:comment "a sensor description for outside temperature
sensor"^^xsd:string ;
    sensorDescription:introduces  outsideTemperature:outsideTemperature ;
    Saref:hasName sensorDescription:T_amb_avg ;
    Saref:hasSingularUnit "?C" ;
    Saref:hasSensorType "Outside temperature" ;
    geo:location <http://Vertigo/floor6/SensorDescription#Vertigo+roof> ;
    Saref:hasCategory sensorDescription:DL_SMS_Mete .

sensorDescription:sensorName
    rdf:type owl:Class ;
    rdfs:comment "the description URIs for sensors"^^xsd:string .

sensorDescription:DT03
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT03"^^xsd:string .

sensorDescription:DT06
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT06"^^xsd:string .

sensorDescription:DT07
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT07"^^xsd:string .

sensorDescription:DT08
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor DT0"^^xsd:string .

sensorDescription:M03
    rdf:type sensorDescription:sensorName ;
```

```
    rdfs:comment "the description URI for sensor M03"^^xsd:string .

sensorDescription:M38
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M38"^^xsd:string .

sensorDescription:M39
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M39"^^xsd:string .

sensorDescription:M40
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M40"^^xsd:string .

sensorDescription:M51
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M51"^^xsd:string .

sensorDescription:M52
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M52"^^xsd:string .

sensorDescription:M53
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor M53"^^xsd:string .

sensorDescription:T_amb_avg
    rdf:type sensorDescription:sensorName ;
    rdfs:comment "the description URI for sensor T_amb_avg"^^xsd:string .

sensorDescription:room
    rdf:type owl:Class;
    rdfs:comment "the room URI where the sensor is located"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Room+6.08>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for room 6.08"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Room+6.18>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for room 6.18"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Room+6.20>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for room 6.20"^^xsd:string .

<http://Vertigo/floor6/SensorDescription#Vertigo+roof>
    rdf:type sensorDescription:room ;
    rdfs:comment "the description URI for Vertigo roof"^^xsd:string .

sensorDescription:category
    rdf:type owl:Class ;
    rdfs:comment "the category for different sensors"^^xsd:string .

sensorDescription:ComfortStatief
    rdf:type sensorDescription:category ;
    rdfs:comments "the log schedule is ComfortStatief"^^xsd:string .

sensorDescription:Modbus
    rdf:type sensorDescription:category ;
    rdfs:comments "the log schedule is Modbus"^^xsd:string .
```

```
sensorDescription:DL_SMS_Mete
    rdf:type sensorDescription:category ;
    rdfs:comments "the log schedule is DL_SMS_Mete"^^xsd:string .

sensor:isSensingValueOf
    rdf:type owl:ObjectProperty ;
    rdfs:comment "connection between sensor values and sensor
representation"^^xsd:string ;
    rdfs:domain sensor:sensorValue ;
    rdfs:range sensor:sensorRepresentation .

sensorDescription:introduces
    rdf:type owl:ObjectProperty ;
    rdfs:comment "connection between sensor description and sensor
representation"^^xsd:string ;
    rdfs:domain sensorDescription:sensorDescription ;
    rdfs:range sensor:sensorRepresentation .

sensorDescription:hasIfcLabel
    rdf:type owl:ObjectProperty ;
    rdfs:comment "connection between sensor room and ifc
space"^^xsd:string ;
    rdfs:domain sensorDescription:room ;
    rdfs:range inst:Ifclabel .
```

Appendix E Python Codes for SPARQL and IfcOpenshell Visualization

```
import rdflib
import time
import math
import ifcopenshell
import ifcopenshell.geom

import numpy as np
import matplotlib.pyplot as plt

# For the colors:
import OCC.Quantity


# read turtle file
model = rdflib.Graph()
model.parse("model.ttl",format="n3")

#get user's topic input
valid1 = False
while valid1 == False:
    topic = raw_input("Please select a topic to query: \n 1. moment CO2, \n
2. moment temperature, \n 3. moment thermal comfort, "
                      "\n 4. seasonal thermal comfort, \n 5. daytime period
thermal comfort\n")
    if topic not in ("moment CO2", "moment temperature", "moment thermal
comfort", "seasonal thermal comfort",
                     "daytime period thermal comfort", "1", "2", "3", "4",
"5"):
        print("Not an appropriate choice. Please select from the three
topics.")
    else:
        valid1 = True

#get user's date input
def getDateInput():
    valid2 = False
    while valid2 == False:
        topicDay = raw_input("Please enter the day you want to query in
\"2015-mm-dd\" format.\n")
        while True:
            try:
                day = time.strptime(topicDay, "2015-%m-%d")
                valid2 = True
                break
            except ValueError :
                print("Invalid format")
                break


    valid3 = False
    while valid3 == False:
        topicTime = raw_input("Please select the time: \"09:30\",
\"13:00\", \"17:30\". \n")
        if topicTime not in ("09:30", "13:00", "17:30"):
            print("Not an appropriate choice. Please select from the three
time choices.")
        else:
            valid3 = True
```

```
    date = "\"" + topicDay + "T" + topicTime+ ":00+01:00\"^^xsd:dateTime"
    return date


# CO2 query
def CO2Monitoring(model):
    date = getDateInput()
    queryCO2 = ( "PREFIX Saref: <https://w3id.org/saref#> " +
                 "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                 "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                 "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                 "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                 "PREFIX  express: <http://purl.org/voc/express#> " +
                 "PREFIX  sensor: <http://www.tue.nl/sensor#> " +
                 "SELECT  ?ID ?value " +
                 "WHERE " +
                 "{ ?a  Saref:hasSensorType  \"CO2\" . " +
                 "  ?a  sensorDescription:introduces   ?b. " +
                 "  ?a  geo:location   ?d. " +
                 "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                 "  ?IfcSpace
ifcowl:longName_IfcSpatialStructureElement   ?IfcLabel . " +
                 "  ?IfcSpace  ifcowl:globalId_IfcRoot  ?e. " +
                 "  ?e  express:hasString  ?ID. " +
                 "  ?c  sensor:isSensingValueOf  ?b. " +
                 "  ?c  owl:hasValue ?value. " +
                 "  ?c  Saref:hasSensingTime  " +
                date +
                "." +
                "}\n" )


    xCO2 = model.query(queryCO2)
    XCO2 = list()
    yCO2 = list()
    resultsCO2 = list()

    for i in xCO2:
        print i

    for i in xCO2:
        XCO2.append(str(i))


    for i in XCO2:
        yCO2.extend(i.split("'"))

    for i in (yCO2[1],yCO2[3],yCO2[8],yCO2[10],yCO2[15],yCO2[17]):
        resultsCO2.append(i)

    print resultsCO2

    # Specify to return pythonOCC shapes from
ifcopenshell.geom.create_shape()
    settings = ifcopenshell.geom.settings()
    settings.set(settings.USE_PYTHON_OPENCASCADE, True)
```

```
    # Initialize a graphical display window
    occ_display = ifcopenshell.geom.utils.initialize_display()

    VertigoFloor6 = ifcopenshell.open(r"Vertigo floor6.ifc")
    products = VertigoFloor6.by_type("IfcProduct")

    guid_to_color = {}

    index = 0
    while index < len(resultsCO2):
        guid = resultsCO2[index]
        value = float(resultsCO2[index + 1])

        clr = (1, 1, 1)

        if value < 600:
            clr = (0, 1, 1)
        elif value < 1000:
            clr = (0, 0, 1)
        elif value < 2500:
            clr = (1, 0, 1)
        else:
            clr = (1, 0, 0)

        clr = OCC.Quantity.Quantity_Color(clr[0], clr[1], clr[2],
OCC.Quantity.Quantity_TOC_RGB)
        guid_to_color[guid] = clr

        index = index + 2

    for product in products:
        if product.Representation:
            shape = ifcopenshell.geom.create_shape(settings,
product).geometry
            clr = guid_to_color.get(product.GlobalId)
            display_shape = ifcopenshell.geom.utils.display_shape(shape,
clr)
            if not clr:

ifcopenshell.geom.utils.set_shape_transparency(display_shape, 0.8)

    occ_display.FitAll()

    ifcopenshell.geom.utils.main_loop()

    return


#inside and outside temperature query
def inAndOutTemperature(model):
    date = getDateInput()
    queryInAndOutTemperature = ("PREFIX Saref: <https://w3id.org/saref#> "
+
                                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                                "PREFIX  owl:
<http://www.w3.org/2002/07/owl#> " +
                                "PREFIX  xsd:
<http://www.w3.org/2001/XMLSchema#> " +
                                "PREFIX  geo:
<http://www.w3.org/2003/01/geo/wgs84_pos#> " +
```

```
                                    "PREFIX  express:
<http://purl.org/voc/express#> " +
                                    "PREFIX  sensor:
<http://www.tue.nl/sensor#> " +
                                    "SELECT  ?ID ?value " +
                                    "WHERE " +
                                    "{ ?a  Saref:hasSensorType  ?type . " +
                                    "  ?a  sensorDescription:introduces   ?b. "
+
                                    "  ?c  sensor:isSensingValueOf  ?b. " +
                                    "  ?c  owl:hasValue ?value. " +
                                    "  ?c  Saref:hasSensingTime  " +
                                    date +
                                    "." +
                                    "  OPTIONAL{?a  geo:location   ?d. " +
                                    "          ?d
sensorDescription:hasIfcLabel   ?IfcLabel. " +
                                    "          ?IfcSpace
ifcowl:longName_IfcSpatialStructureElement   ?IfcLabel . " +
                                    "          ?IfcSpace
ifcowl:globalId_IfcRoot  ?e. " +
                                    "          ?e  express:hasString   ?ID .} "
+
                                    "  FILTER ((?type = \"Air temperature at
1.1 m\") || (?type = \"Temperature\") || (?type = \"Outside
temperature\"))" +
                                    "}\n" +
                                    "ORDER BY ?ID")


    xInAndOut = model.query(queryInAndOutTemperature)
    XInAndOut = list()
    yInAndOut = list()
    resultsInAndOut = list()

    for i in xInAndOut:
        print i

    for i in xInAndOut:
        XInAndOut.append(str(i))

    for i in XInAndOut:
        yInAndOut.extend(i.split("'"))

    yInAndOut[0] = "2tv2ExGFr31fuFHLjs39yh"

    for i in (
    yInAndOut[0], yInAndOut[1], yInAndOut[6], yInAndOut[8], yInAndOut[13],
yInAndOut[15], yInAndOut[20], yInAndOut[22]):
        resultsInAndOut.append(i)

    print resultsInAndOut

    # Specify to return pythonOCC shapes from
ifcopenshell.geom.create_shape()
    settings = ifcopenshell.geom.settings()
    settings.set(settings.USE_PYTHON_OPENCASCADE, True)

    # Initialize a graphical display window
    occ_display = ifcopenshell.geom.utils.initialize_display()
```

```
    VertigoFloor6 = ifcopenshell.open(r"Vertigo floor6.ifc")
    products = VertigoFloor6.by_type("IfcProduct")

    guid_to_color = {}

    index = 0
    while index < len(resultsInAndOut):
        guid = resultsInAndOut[index]
        value = float(resultsInAndOut[index + 1])

        clr = (1, 1, 1)

        if value < 20:
            clr = (0, 1, 1)
        elif value < 26:
            clr = (0, 1, 0)
        elif value < 30:
            clr = (1, 1, 0)
        else:
            clr = (1, 0, 0)

        clr = OCC.Quantity.Quantity_Color(clr[0], clr[1], clr[2],
OCC.Quantity.Quantity_TOC_RGB)
        guid_to_color[guid] = clr

        index = index + 2

    for product in products:
        if product.Representation:
            shape = ifcopenshell.geom.create_shape(settings,
product).geometry
            clr = guid_to_color.get(product.GlobalId)
            display_shape = ifcopenshell.geom.utils.display_shape(shape,
clr)
            if not clr:

ifcopenshell.geom.utils.set_shape_transparency(display_shape, 0.8)

    occ_display.FitAll()

    ifcopenshell.geom.utils.main_loop()
    return


#sample thermal comfort query:
def thermalComfort(model):
    date = getDateInput()
    # air speed query
    queryAirSpeed = ( "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "PREFIX  sensor: <http://www.tue.nl/sensor#> " +
                "SELECT  ?ID ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Air speed at 1.1 m\" . " +
                "  ?a  sensorDescription:introduces   ?b. " +
```

```
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcSpace
ifcowl:longName_IfcSpatialStructureElement   ?IfcLabel . " +
                "  ?IfcSpace  ifcowl:globalId_IfcRoot  ?e. " +
                "  ?e  express:hasString  ?ID. " +
                "  ?c  sensor:isSensingValueOf  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime  " +
                date +
                "." +
                "}\n" )


    xAirSpeed = model.query(queryAirSpeed)
    XAirSpeed = list()
    yAirSpeed = list()
    resultsAirSpeed = list()

    for i in xAirSpeed:
        print i

    for i in xAirSpeed:
        XAirSpeed.append(str(i))


    for i in XAirSpeed:
        yAirSpeed.extend(i.split("'"))

    for i in (yAirSpeed[1], float(yAirSpeed[3])):
        resultsAirSpeed.append(i)

    print "air speed: " , resultsAirSpeed

    #air temperature query
    queryAirTemperature = ( "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "PREFIX  sensor: <http://www.tue.nl/sensor#> " +
                "SELECT  ?ID ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Air temperature at 1.1 m\" .
" +
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcSpace
ifcowl:longName_IfcSpatialStructureElement   ?IfcLabel . " +
                "  ?IfcSpace  ifcowl:globalId_IfcRoot  ?e. " +
                "  ?e  express:hasString  ?ID. " +
                "  ?c  sensor:isSensingValueOf  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime  " +
                date +
                "." +
                "}\n" )
```

```
    xAirTemperature = model.query(queryAirTemperature)
    XAirTemperature = list()
    yAirTemperature = list()
    resultsAirTemperature = list()

    for i in xAirTemperature:
        print i

    for i in xAirTemperature:
        XAirTemperature.append(str(i))


    for i in XAirTemperature:
        yAirTemperature.extend(i.split("'"))

    for i in (yAirTemperature[1], float(yAirTemperature[3])):
        resultsAirTemperature.append(i)

    print "temperature " , resultsAirTemperature


    #temperature black globe query
    queryBlackGlobe = ( "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "PREFIX  sensor: <http://www.tue.nl/sensor#> " +
                "SELECT  ?ID ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Temperature black globe at
0.6 m\" . " +
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcSpace
ifcowl:longName_IfcSpatialStructureElement   ?IfcLabel . " +
                "  ?IfcSpace  ifcowl:globalId_IfcRoot  ?e. " +
                "  ?e  express:hasString  ?ID. " +
                "  ?c  sensor:isSensingValueOf  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime  " +
                date +
                "." +
                "}\n" )


    xBlackGlobe = model.query(queryBlackGlobe)
    XBlackGlobe = list()
    yBlackGlobe = list()
    resultsBlackGlobe = list()

    for i in xBlackGlobe:
        print i

    for i in xBlackGlobe:
```

```
        XBlackGlobe.append(str(i))


    for i in XBlackGlobe:
        yBlackGlobe.extend(i.split("'"))

    for i in (yBlackGlobe[1], float(yBlackGlobe[3])):
        resultsBlackGlobe.append(i)

    print "black globe:  " , resultsBlackGlobe

    #relative humidity query
    queryHumidity = ( "PREFIX Saref: <https://w3id.org/saref#> " +
                "PREFIX sensorDescription:
<http://Vertigo/floor6/SensorDescription#> " +
                "PREFIX  owl: <http://www.w3.org/2002/07/owl#> " +
                "PREFIX  xsd: <http://www.w3.org/2001/XMLSchema#> " +
                "PREFIX  geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> "
+
                "PREFIX  express: <http://purl.org/voc/express#> " +
                "PREFIX  sensor: <http://www.tue.nl/sensor#> " +
                "SELECT  ?ID ?value " +
                "WHERE " +
                "{ ?a  Saref:hasSensorType  \"Relative Humidity at 0.6
m\" . " +
                "  ?a  sensorDescription:introduces   ?b. " +
                "  ?a  geo:location   ?d. " +
                "  ?d  sensorDescription:hasIfcLabel   ?IfcLabel. " +
                "  ?IfcSpace
ifcowl:longName_IfcSpatialStructureElement   ?IfcLabel . " +
                "  ?IfcSpace  ifcowl:globalId_IfcRoot  ?e. " +
                "  ?e  express:hasString  ?ID. " +
                "  ?c  sensor:isSensingValueOf  ?b. " +
                "  ?c  owl:hasValue ?value. " +
                "  ?c  Saref:hasSensingTime  " +
                date +
                "." +
                "}\n" )


    xHumidity = model.query(queryHumidity)
    XHumidity = list()
    yHumidity = list()
    resultsHumidity = list()

    for i in xHumidity:
        print i

    for i in xHumidity:
        XHumidity.append(str(i))


    for i in XHumidity:
        yHumidity.extend(i.split("'"))

    for i in (yHumidity[1],float(yHumidity[3])):
        resultsHumidity.append(i)

    print "humidity " , resultsHumidity
```

```
    #calculate PMV and PPD
    m = 70.0
    w = 0.0
    pw = 0.0
    if(resultsAirTemperature[1] >= 18 and resultsAirTemperature[1] < 21):
        pw = (resultsAirTemperature[1] - 18) / 3 * 4 + 18
    elif (resultsAirTemperature[1] >= 21 and resultsAirTemperature[1] <
24):
        pw = (resultsAirTemperature[1] - 21) / 3 * 4.6 + 21
    elif (resultsAirTemperature[1] >= 24 and resultsAirTemperature[1] <
27):
        pw = (resultsAirTemperature[1] - 24) / 3 * 5.4 + 24
    elif (resultsAirTemperature[1] >= 27 and resultsAirTemperature[1] <
29):
        pw = (resultsAirTemperature[1] - 27) / 3 * 6 + 27

    pa = resultsHumidity[1] / 1000 * pw
    hc = 12.1 * math.pow(resultsAirSpeed[1], 0.5)
    icl = 1.0
    rcl = 0.155 * icl
    fcl = (1.0 + 0.2 * icl) / (1.05 + 0.1 *icl)
    tcl = 35.7 - 0.0275 * (m - w) - rcl * ((m - w) - 3.05 * (5.73 - 0.007 *
(m - w) - pa) -
          0.42 * ((m - w) - 58.15) - 0.0173 * m * (5.87 - pa) - 0.0014 * m
* (34 - resultsAirTemperature[1]))
    tr = math.pow(math.pow((resultsBlackGlobe[1] + 273), 4) + 2.5 *
math.pow(10, 8) * math.pow(resultsAirSpeed[1], 0.6) *
        (resultsBlackGlobe[1] - resultsAirTemperature[1]), 0.25) - 273
    PMV = (0.303 * math.exp(-0.036 * m) + 0.028) * ((m - w) - 3.96e-8 * fcl
* (math.pow((tcl + 273), 4) -
          math.pow((tr + 273), 4)) - fcl * hc * (tcl -
resultsAirTemperature[1]) - 3.05 * (5.73 - 0.007 * (m - w) - pa) -
          0.42 * ((m - w) - 58.15) - 0.0173 * m * (5.87 - pa) - 0.0014 * m
* ( 34 - resultsAirTemperature[1]))
    PPD = 100 - 95 * math.exp(-(0.3353 * math.pow(PMV, 4) + 0.2179 *
math.pow(PMV, 2)));

    print "PMV: " , PMV , "   PPD: " , PPD

    # Specify to return pythonOCC shapes from
ifcopenshell.geom.create_shape()
    settings = ifcopenshell.geom.settings()
    settings.set(settings.USE_PYTHON_OPENCASCADE, True)

    # Initialize a graphical display window
    occ_display = ifcopenshell.geom.utils.initialize_display()

    VertigoFloor6 = ifcopenshell.open(r"Vertigo floor6.ifc")
    products = VertigoFloor6.by_type("IfcProduct")

    guid_to_color = {}

    guid = resultsAirSpeed[0]
    value = PPD

    clr = (1, 1, 1)

    if value < 25:
        clr = (0, 1, 1)
    elif value < 50:
        clr = (0, 0, 1)
```

```
    elif value < 75:
        clr = (1, 0, 1)
    else:
        clr = (1, 0, 0)

    clr = OCC.Quantity.Quantity_Color(clr[0], clr[1], clr[2],
OCC.Quantity.Quantity_TOC_RGB)
    guid_to_color[guid] = clr

    for product in products:
        if product.Representation:
            shape = ifcopenshell.geom.create_shape(settings,
product).geometry
            clr = guid_to_color.get(product.GlobalId)
            display_shape = ifcopenshell.geom.utils.display_shape(shape,
clr)
            if not clr:

ifcopenshell.geom.utils.set_shape_transparency(display_shape, 0.8)

    occ_display.FitAll()

    ifcopenshell.geom.utils.main_loop()

    return

def seasonalThermalComfort():
    print("The average predicted percentage of dissatisfied occupants \nfor
February is 13.17%, \nfor April is"
          " 39.82%, \nfor July is 89.78%, \nand for October is 11.25%. \n")
    #create histogram
    month = ['February', 'April', 'July', 'October']
    percentage = [(13.17 / 100), (39.82 / 100), (89.78 / 100), (11.25 /
100)]

    pos = np.arange(len(month))
    width = 1.0

    ax = plt.axes()
    ax.set_xticks(pos + (width / 2))
    ax.set_xticklabels(month)

    ax.set_ylim([0, 1])

    ax.set_xlabel('month')
    ax.set_ylabel('PPD')

    plt.bar(pos, percentage, width, color='g')
    plt.show()
    return

def daytimeThermalComfort():
    print("The average predicted percentage of dissatisfied occupants for
April \n at 09:30 is 49.54%, \n at 13:00 is "
          "39.82%, \n at 17:30 is 37.00%. \n")
    #create histogram
    month = ['09:30', '13:00', '17:30']
    percentage = [(49.54 / 100), (39.82 / 100), (37.00 / 100)]

    pos = np.arange(len(month))
    width = 1.0
```

```
    ax = plt.axes()
    ax.set_xticks(pos + (width / 2))
    ax.set_xticklabels(month)

    ax.set_ylim([0, 1])

    ax.set_xlabel('time')
    ax.set_ylabel('PPD')

    plt.bar(pos, percentage, width, color='g')
    plt.show()
    return


#judge for the topic
if topic == "moment CO2" or topic == "1" :
    CO2Monitoring(model)
elif topic == "moment temperature" or topic == "2" :
    inAndOutTemperature(model)
elif topic == "moment thermal comfort" or topic == "3":
    thermalComfort(model)
elif topic == "seasonal thermal comfort" or topic == "4":
    seasonalThermalComfort()
elif topic == "daytime period thermal comfort" or topic == "5":
    daytimeThermalComfort()
```