Addressing the limited adoption of Semantic Web for data integration in AEC industry: Exploration of an RDF-based approach depending on available ontologies and literals matching

Dimitrios Patsias*

University of Twente, Faculty of Engineering Technology, P.O. Box 217, 7500 AE Enschede, The Netherlands
**Corresponding author: dimitrispatsias@gmail.com*

A B S T R A C T

The limitations of Building Information Modeling (BIM) regarding the complete and integrated management support for building data, which are vast, diverse and distributed across different sources, systems and actors and which often need to be combined for the purpose of several analyses, stimulated the use of Semantic Web. The latter enables the uniform representation of heterogeneous data in structured graphs, using the Resource Description Framework (RDF) and common languages that describe relationships among these, namely ontologies. By deploying Semantic Web technologies, several research streams have focused on the integration of BIM with cross-domain data such as GIS, product and material data, sensor and cost data among others, by suggesting the development of several ontologies that represent concepts and relationships from different domains and the establishment of semantic links between them. At the same time, people without prior relevant awareness, perceive the concept of ontologies and tasks related to them as something too complicated. This is considered the main reason for the slow adoption of Semantic Web technologies, which is obvious within the AEC-FM industry as well. In this paper, the feasibility of a data integration approach that uses available ontologies and avoids ontology alignment is explored. For that reason, an approach based on the RDF representation of diverse datasets, the semantic description of these using available ontologies and the integration of these by matching literals between datasets, instead of establishing semantic links, is presented. A conceptual framework is developed and tested in the context of a simple but typical scenario where BIM has to be integrated with heterogeneous data sources in order to perform several analyses. The results of the evaluation showed that although proven successful for linking datasets, the unlikelihood of the existence of ontologies that fully describe the datasets and cover the data needs of every organization, the requirements regarding the lexical and data type similarity of the literals, the required prior specific knowledge about the data and the resulting complexity of the queries, render this approach hard to be properly applied and generalized. The discussion and conclusion sections, elaborate on the tradeoffs of the approach and the lessons learned regarding the requirements, the potential and the challenges for engineering companies in their effort to comprehend and successfully deploy Semantic Web technologies within their everyday data management practices.

Keywords: Semantic Web, BIM, data integration, RDF, AEC

1. Introduction

The Architecture, Engineering, Construction and Facilities Management (AEC-FM) industry is a knowledge-intensive one and currently, an enormous growth is experienced in its capacity to both generate and collect data. As the construction industry is adapting to new computer technologies, more and more computerized construction data are becoming available (Soibelman, Wu, Caldas, Brilakis, & Lin, 2008). Moreover, various forms of knowledge and data, stemming from multiple disciplines within the construction industry have to be combined in order to facilitate the communication and collaboration among several stakeholders and therefore enhance the success of construction projects. This increasing data flow has increased the complexity of decision-making. Furthermore, accessing this data and integrating all this information is a difficult challenge as it is often stored in disconnected software applications and databases.

In recent years, Building Information Modeling (BIM) technologies have significantly influenced information management practices throughout the entire building industry. BIM has been established as a concept of using IT to model and manage data and has demonstrated the potential for tackling problems stemming from insufficient access to information in all phases of a construction project (Parsanezhad & Dimyadi, 2014). A

Building Information Model can be seen as a central repository of building data that can be used by all project stakeholders (Curry et al., 2013). In order to increase interoperability throughout the design and construction process within the building design team, the buildingSMART organization developed and maintained the Industry Foundation Classes (IFC) standard. The IFC data model is represented as a schema in the expressive EXPRESS data specification language and aims at providing a central "conceptual data schema and an exchange file format for BIM data" (Liebich et al., 2013). By exporting a native BIM format, which varies depending on the respective software, to the neutral IFC format or the other way around, BIM data can be exchanged between heterogeneous software applications, covering a wide range of use cases including 4D planning, 5D cost calculation and structural analysis (Pauwels, Krijnen, Terkaj, & Beetz, 2017). Many national BIM standards strongly recommend or even impose the use of IFC to enhance the exchange and communication of building information among project stakeholders.

However, despite the advantages in terms of software interoperability, within the wider context of an organization, BIM models are only silos of information (Curry et al., 2013) and there is a need for utilizing other relevant data generated outside of the context of building projects as well in order to perform various analyses. To date, however, linking of building data stored in an IFC file, which mainly concern a building's geometry, with other relevant data such as, for example, regulations, GIS, building products and materials and sensor data, is limitedly possible. Data stemming from different domains have heterogeneous data formats and their integration is not easy because each domain develops its own data schema to represent properties for the same object (Niknam & Karshenas, 2017). Therefore, combining data about a building element requires accessing several project documents and manually extracting, combining, validating and sometimes inferring the necessary information for every kind of analysis, which is a cumbersome and error-prone process (Gonçal Costa, 2017). A typical example is during the development of a 3D building model, when the modeler has to assess several types of documents to make sure that the building complies to relevant regulations or that the most suitable building elements have been selected. These documents can be quite large and although it is possible to attach them to BIM elements (e.g. as pdf or excel spreadsheets), attached documents are hard to be queried by computers to find relevant data for different computing tasks (Niknam, 2017). Therefore, the user has to identify and interpret the relevant information as well as implicitly relate it with BIM elements.

The need to address the aforementioned issue, among others, has stimulated the usage of Semantic Web technologies in the domains of AEC-FM over the recent years and they are typically considered as complementary to Building Information Modelling software, focusing on organizing and sharing the 'semantics' of a building and not only on the display of its geometric aspects (Pauwels, Zhang, & Lee, 2017). Semantic Web technologies, enable the representation of different heterogeneous data into a common standard, namely the Resource Description Framework (RDF) (Schreiber & Raimond, 2014) and the semantic description of these and their relationships due to the expressive power of Ontology Web Languages (OWL) (Hitzler, Krötzsch, Parsia, & Sebastian, 2012). They make it possible to refer to a specific piece of information contained in a document or program and allow to represent it in structured graphs and efficiently integrate building information of completely different nature (Pauwels, Zhang, et al., 2017). Several research initiatives have addressed the integration of BIM with diverse external data by making use of Semantic Web technologies.

The Semantic Web approaches adopted in the academic world usually include the representation of diverse data as RDF graphs and the development of several ontologies that represent different domains by modeling the data and describing concepts, instances, and relationships among them. Ontologies have a central role in Semantic Web-based data integration which is enabled through the definition of mappings and links between data, which define binary relations between entities that belong to different datasets and relate semantically one to another (Ferrara & Nikolov, 2011). This way, information of the data sources can be integrated to allow agents (human and machines) have a more unified view.

However, the discovery and establishment of these mappings, which is called ontology alignment, is considered a challenging task that requires collaboration between ontology engineers and domain experts (Gonçal Costa, 2017), especially when the datasets are large and complex (Cheatham & Pesquita, 2017). Finding correspondences between entities included in different ontologies is very subjective and depends on the way in which reality is represented in each one of them. Therefore, human interpretation plays an essential role. Pauwels, Zhang, et al. (2017), in their literature overview about semantic web technologies in AEC industry, highlighted the lack of sufficient explanation about the way that semantic links between ontologies are defined and managed. This aspect, according to them, is considered as one of the key research challenges for the implementation of

Semantic Web technologies in the construction industry, which so far is relatively slow (Godager, 2018), since the lack of a systematic and elaborate explanation of this mapping process hinders comprehension and correct implementation. Regarding the concept of ontologies in general, Kuriakose (2009) stated that one of the major challenges in Semantic Technology deployment is the long learning curve regarding ontology development, which is a difficult and time-consuming process. This comes in accordance with Lanthaler & Gütl (2011) and Serrano & Stroulia (2017) who noticed that most web developers have a reluctance towards Semantic Web, called *Semaphobia* and they perceive its technologies, and especially the concept of ontologies, as something too complex. According to Garshol (2013), ordinary data modeling being already considered complicated and tricky by many developers, it can be understood that modeling in description logics, which are families of formal knowledge representation languages on which ontologies are based, can be rather frightening to them.

Taking the above into account, and given the fact that there are many actors in the construction industry with limited knowledge about how to exploit Semantic Web technologies (Godager, 2018), the desired integration of BIM with external relevant building data, requires rather hard work by AEC practitioners, in terms of comprehending the concepts of semantic data modelling and ontologies, developing ontologies that efficiently describe their data and cover their data needs, and establishing semantic links between them, so that the aforementioned integration is possible. This complexity, in combination with the lack of sufficient documentation and explanation of tasks that require high amounts of human interpretation, such as ontology alignment, places the threshold for Semantic Web adoption by construction companies rather high and renders the step towards these technologies difficult. In an attempt to deal with this, this paper claims that alternative approaches could be explored, where the desired data integration could be achieved in a more familiar to the user way, based on practices that could be easier comprehended and applied.

In this regard, this paper explores an approach for RDF-based integration of BIM with relevant data which depends on the usage of existing ontologies and on literals[1] matching between different datasets. While suggesting the usage of available ontologies in order to avoid the challenging task of ontology development, the scope of this study addresses mainly the topic of ontology mapping, by exploring an alternative, potentially easier to comprehend and apply, way of interacting between different datasets and combining data included in these. Instead of identifying and establishing semantic links between entities, which is rather challenging and requires reflection and collaborative work by ontology engineers and domain experts, this is done by matching values of instances in the datasets and then, based on whether this matching is true or not, retrieving relevant data from the datasets. The proposed conceptual framework constitutes a step-wise guideline for activities and considerations that are required from engineering companies in order to integrate BIM with relevant data. With the use of a typical scenario, the applicability and feasibility of the framework are tested. More specifically, the benefits, the generalizability, as well as the requirements and the restrictions of the approach, are evaluated. The discussion and conclusion derived elaborate on the tradeoffs of the approach and the lessons learned regarding the requirements, the potential and the challenges for engineering companies in their effort to comprehend and successfully deploy Semantic Web technologies within their everyday data management practices.

In section 2, an overview of the basic technologies of the Semantic Web is presented in order to provide a basic understanding of their essence and functionality and some representative research approaches related to the integration of BIM with cross-domain data are mentioned. Sections 4 and 5 present the proposed approach and the proof of concept respectively. Finally, the evaluation of the approach is discussed and some concluding remarks and recommendations are highlighted in sections 6 and 7.

2. Semantic Web technologies

Tim Berners-Lee (1998), director of the World Wide Web Consortium (W3C) first defined the term "Semantic Web" as "the merging of human-readable documents with machine-understandable data", which implies that the data is not only machine-readable but can also be interpreted or 'understood' correctly by computers. In his publication in Scientific American (2001), Berners-Lee further clarifies the concept of the Semantic Web as "an extension of the current web in which information is given well-defined meaning, better-enabling computers and people to work in cooperation". In fact, the Semantic Web makes it possible to refer to a

[1] In computer science, a literal is a notation for representing a fixed value in source code. There are five types of literals: integer, floating point, boolean, character, and string (Wikipedia).

specific piece of information contained in a document or program, instead of having to connect to the document or program themselves. If this information is updated, you can take advantage of this update (Godager, 2018). Therefore, it could be said that it enables the transition from the traditional web (Web of Documents) to a Web of Data.

In this section, the most essential Semantic Web technologies namely RDF, RDFS, OWL and SPARQL are presented in detail, followed by a representative sample of research initiatives, where these technologies were deployed to achieve integration of BIM with cross-domain data.

2.1 Resource Description Framework (RDF)

The Resource Description Framework (RDF) is a flexible and generic standard that allows representation and integration of data from diverse domains. It is a data model designed to represent information stemming from multiple sources, being heterogeneously structured and having different schemata in an integrated and unified way (Heath & Bizer, 2011). Data that is stored based on the RDF is represented by triples that consist of a subject, an object and a predicate, which describes the type of relationship between those two. All three of them are encoded as URIs (Universal Resource Identifiers), which provide a basic mechanism to globally and uniquely identify resources and form the basic mechanism used to associate disjoint pieces of data. An example of this logic is shown in **Fig. 1** below, where the triple consists of the Netherlands, which is the subject, Amsterdam, which is the object, and the predicate which defines the relationship between those two, namely that Amsterdam is the capital of Netherlands.

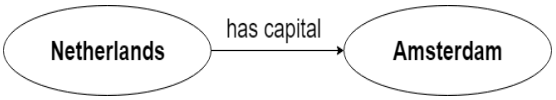

Fig. 1. Visual representation of an RDF triple

The same resource can often be referenced in multiple triples. This ability to have the same resource be in the subject position of one triple and the object position of another makes it possible to find connections between triples, which is an important part of RDF's power (Schreiber & Raimond, 2014). These triples can be visualized as a connected graph consisting of nodes (made up by the subjects and the objects) and arcs (formed by the predicates). All the above can be better understood with the use of **Fig. 2**. In this example, Netherlands is the subject of four triples, and Amsterdam is the subject of one and the object of two triples. Therefore, the graph created can provide diverse information about resources. Each node in such a graph represents a concept or object in the world. By describing all information as such in interlinked RDF graphs, a uniform representation of information is achieved, making information reusable by both humans and computer applications (Pauwels, Zhang, et al., 2017).

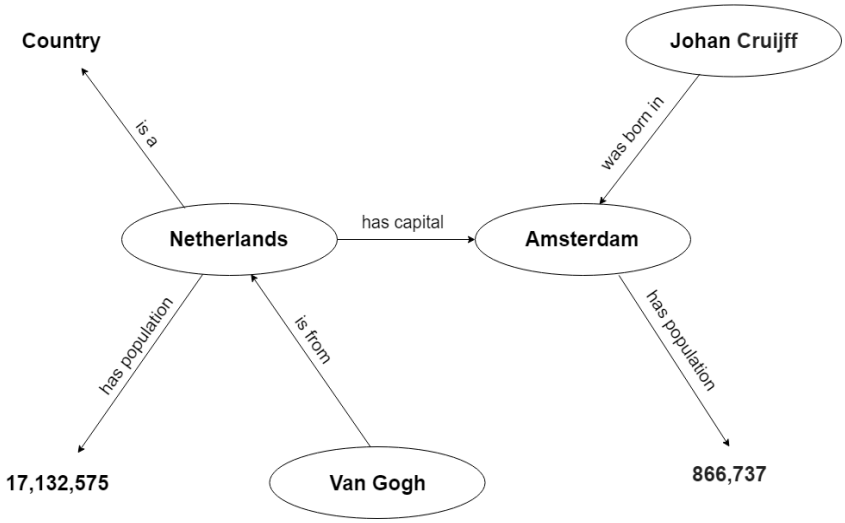

Fig. 2. Graph of RDF triples

RDF links entities within documents and enables the user to state explicitly the nature of this connection, so that these links can be followed by applications to discover more data. Therefore, publishing and linking data using RDF contributes to a Web where data is significantly more discoverable, and therefore more usable (Heath & Bizer, 2011). Such uses of RDF are often referred to as Linked Data, which is "a set of techniques to represent and connect structured data on the web" (Berners-Lee, 2006; Wood, Zaidman, Ruth, & Hausenblas, 2014).

Before publishing an RDF graph on the Web, it must first be serialized using an RDF syntax. This means that particular syntax is used to write the triples that turn an RDF graph into a file (Heath & Bizer, 2011). Some popular RDF serialization formats are Turtle, N-Triples, RDF/XML and JSON-LD.

A collection of multiple RDF graphs is called an RDF dataset and larger datasets are often stored in specialized databases called triple stores, that can be accessed and queried over regular network structures (Pauwels, McGlinn, Törmä, & Beetz, 2018).

2.2 RDF Schema and Web Ontology Language (OWL)

RDF does not provide any domain-specific terms that describe formal hierarchies of things in the world and how they relate to each other. This function is served by vocabularies and ontologies, which can be described as knowledge structures that create frameworks of relevant concepts and connections between them. The most basic elements describing such ontologies are contained in the RDF Schema (RDFS) vocabulary, which consists of the specifications of classes, subclasses, comments, and data types (Pauwels, Zhang, et al., 2017). A much larger vocabulary to describe ontologies are provided within Web Ontology Language (OWL) which "is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things". OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit (Hitzler et al., 2012)". The "**has capital**", **"has population", "was born in"** statements in **Fig. 2.** above are such examples.

2.3 The Simple Protocol and RDF Query Language (SPARQL)

Using the RDF, there is a possibility of querying the information captured in RDFS/OWL. The primary query language for RDF graphs is SPARQL, or the Simple Protocol and RDF Query Language, which is able if defined correctly, to retrieve, create, update or delete such information. SPARQL is one of the essential technologies of the Semantic Web and it works by using a triple pattern including both resources and variables that are being matched against a data triple. Searching with a SPARQL query is actually matching patterns in the query to patterns in the dataset (Petkova, 2019). The triple pattern is to specify what information needs to be taken from the triple and how the entities and variables relate to each other in it. A basic query in SPARQL is built like this:

```
<prefixes> SELECT <variables> WHERE <pattern>
```

where **<prefixes>** indicate vocabularies that are used with their abbreviations during the query, **<variables>** store the results that should be returned and **<pattern>** is the pattern in the graph that should be matched. An example is provided in **Fig. 2** below, where it is requested to get the names of all people who know Bob White. The vocabulary used is indicated, namely FOAF (Friend Of A Friend)[2], **?s** is the variables' results that will be returned and the pattern **?s foaf:knows "Bod White"** is the pattern that should be matched in order to get all the people that know Bod White.

[2] See : http://xmlns.com/foaf/spec/

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?s
WHERE  {
  ?s  foaf:knows  "Bob White".
}
```

Fig. 2. SPARQL example

By making use of statements such as `FILTER, GROUP BY, ORDER BY, COUNT,` etc. among others[3], it is possible to apply restrictions to the solutions, calculate aggregate values, establish the order of a solution sequence, create summary values for groups of triples, etc. This way the results can be tailored to the specific information needs of the user.

SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. From 2013, the `SERVICE` extension to SPARQL 1.1, allows to direct a portion of a query to a particular SPARQL endpoint and then the generated results are combined with results from the rest of the query (Prud'hommeaux & Buil-Aranda, 2013). This way, it is made possible to combine data from resources residing in distant repositories.

2.4. Related work on the integration of BIM with cross-domain data

By taking advantage of the possibilities that the aforementioned Semantic Web technologies can offer regarding data integration, among others, several research initiatives have addressed the issue of integrating BIM with cross-domain data. In this section, a representative sample is presented in order to highlight the most common practices.

Niknam & Karshenas (2017) suggested a shared ontology approach for the semantic representation of cross-domain building information. They organized concepts common to all building lifecycle domains in a uniform classification system and defined a BIM-shared ontology that describes design properties in a knowledge base. Then, cost and schedule ontologies were developed by extending the shared ontology and SPARQL queries provided information by integrating these, using the shared ontology as a query mediator.

Hor, Jadidi, & Sohn (2016) addressed the integration of BIM with GIS by suggesting the development of two ontologies that describe IFC and GIS data respectively and a resulting integrated ontology by linking similar concepts and relationships between them, using a mapping algorithm. Querying the integrated graph provided a unified view over an integrated knowledge base, combining BIM and GIS data.

Costa & Madrazo (2015), as part of the research project BAUKOM, created a semantic catalog of building components which were described based on an ontology that they developed. Their purpose was to examine the components used on a BIM model and provide compatible products extracted from the catalog, based on rules of semantic inference, according to safety and structural regulations, which were represented in the RDF.

Lee, Kim, & Yu (2014), suggested that the search for the most suitable building elements and materials for the required construction works can be automated. For that reason, using the example of tiles, they propose the translation of BIM data to RDF, the extraction of work conditions such as building element, finishing base type and finishing thickness (input) out of these and work items that can be inferred from semantic reasoning rules (output), according to two domain ontologies, the Tiling Work Condition Ontology (TWCO) and the Tiling Work Item Ontology (TWIO). As a result, the most appropriate materials can be selected, enabling accurate and reliable cost estimation results.

The above-described researches address diverse domains and their integration with BIM and the need for ontology development and ontology alignment is their common denominator. By developing ontologies that describe datasets in an efficient way, based on the purpose of the application, and by establishing semantic links between related entities described by different ontologies, they prove that data integration is possible.

[3] See : https://www.w3.org/TR/sparql11-query/

However, although creating RDF graphs, linking and using them for querying and reasoning is explained, explanation about the decisions to be made in order to link different datasets and ontologies is usually missing (Pauwels, Zhang, et al., 2017). These tasks, which depend on human interpretation, require deep knowledge of the domain in use as well as of ontology engineering and the lack of documentation hinders the reader from comprehending how they are implemented. The lack of documentation was highlighted by Godager (2018) in his literature review of the integration of BIM with Semantic Web technology, where he concluded that researches that are open and accessible to users in the BIM field are very few.

3. RDF-based integration, depending on literals matching

The current research, having acknowledged the complexity of semantic data modeling and ontologies for the average AEC practitioner and the lack of sufficient explanation and documentation, which inhibits comprehension and applicability, explores an alternative and potentially easier to implement and adopt approach for integrating BIM with cross-domain data. This section presents an approach based on the RDF representation of diverse datasets, the semantic description of these using available ontologies and the integration of these by matching literals between datasets, instead of establishing semantic links. The scope of this approach is limited to internal data management tasks of a construction company that require the combination of BIM models with relevant data and the extraction of useful information to perform various decision analyses.

In order to develop a conceptual framework for the approach, the "Guidelines for Linked Data generation and publication" (Radulovic et al., 2015) have been consulted and adjusted based on the scope and the needs of the current research, namely the integration of BIM with relevant building data based on RDF and literals matching. This is a detailed set of guidelines, including necessary steps, tools, alternatives, examples and recommendations, focusing specifically on construction industry use cases and recommended by several researchers (Chou & Ngo, 2016; Cuenca, Larrinaga, & Curry, 2017; Kalpoe, 2016; Pauwels, Zhang, et al., 2017; Yu, 2016). Although Radulovic's guidelines concern the generation and publication of energy consumption data as Linked Data on the web and include tasks that are outside of the scope of the current paper, such as ontology development, data linking and publication, several essential tasks of the process were utilized and adjusted in order to fit the purposes of the current study.

The developed framework is presented in **Fig. 3** below. The whole process includes five main tasks and each task is divided into several steps which will be introduced in the following sections.

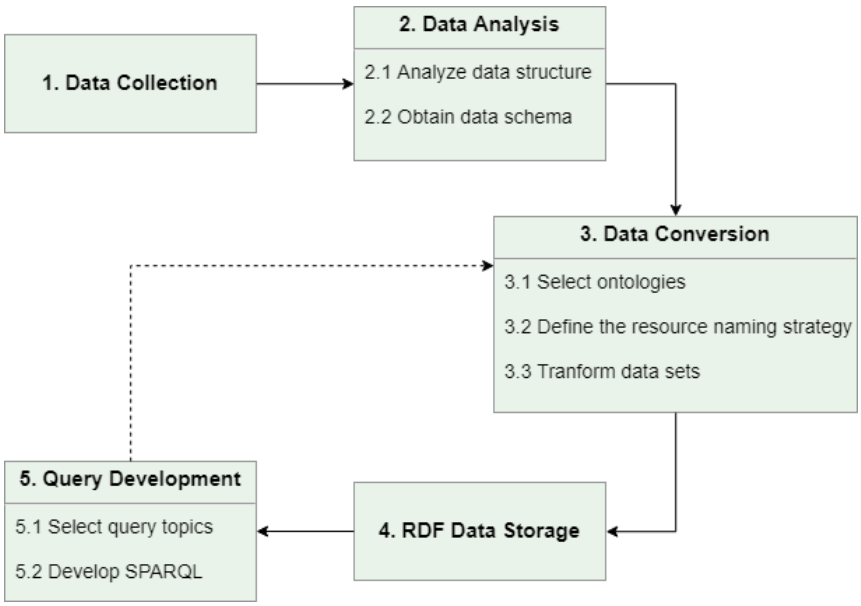

Fig. 3. Conceptual Framework

3.1 Data collection

The first task of the process concerns the collection of the data that are to be transformed into RDF, namely the BIM model and the relevant data that it is to be integrated with. These data might be owned by the company or provided by a collaborating partner.

3.2 Data analysis

Once the data have been obtained, the next task is to look into the data themselves.

3.2.1 Analyze data structure

The first step is to analyze how data are structured and organized. The type of information that they provide has to be understood and the level to which they are structured has to be defined. Unstructured data are more difficult to use because they require a level of structuring first, therefore transformations might be required in order to facilitate the following data processing.

3.2.2 Obtain data schema

Obtaining data schema aims at identifying the domain concepts that are described in the data set, as well as the relations among them. In case the schema is not available, it has to be extracted from the data, meaning that the user has to identify what the provided data represent and how they are related to each other. This step is very important regarding the suggested integration approach, which is based on literals matching. Therefore, the existence of common concepts in both datasets is essential for the interaction between them.

3.3. Data conversion

This task concerns the transformation of input data into the unified RDF format, in order to enable interaction between originally heterogeneous datasets. It includes three main steps.

3.3.1 Select ontology

One of the most essential tasks of the whole process is to select an ontology that describes and represents the terms, concepts, and relationships of the data. This way, not only will the data be semantically enriched, but it will also be possible to identify patterns within the datasets and match them with query patterns in order to obtain specific information. As explained above, the purpose of this research is to explore the challenges of implementing an RDF-based integration depending on available ontologies. Therefore, a prerequisite of this approach is to make use of existing ontologies. The domain of use, the purpose and the requirements that it should cover, should be taken into account before selecting an ontology since there can be ontologies that describe the same domain but have been developed having different scopes and purposes.

There are plenty ontologies that cover a wide range of domains and the W3C (World Wide Web Consortium) suggests a list of them[4], while the Linked Open Vocabulary (LOV) is a high-quality catalog of reusable vocabularies for the description of data semantics (Vandenbussche, Atemezing, & Vatant, 2014).

3.3.2 Define the resource naming strategy

One of the basic principles of RDF is the use of Uniform Resource Identifiers (URIs) to denote instances so that data become unique. Radulovic et al. (2015) suggest consulting well-established guidelines when designing URIs (the W3C working group has provided a detailed list of such guidelines (Hyland, Atemezing, & Villazón-Terrazas, 2014)) and they divide the process of defining a resource naming strategy in three sub-steps.

The first one concerns the choice of a URI form. It can be either a hash URI, in which a fragment that is separated from the rest of the URI by a hash character ('#') is contained or a slash URI ('/'). The second one is to choose a domain and a path that form the base URI and the third sub-step is to choose a pattern for ontology classes, properties, and individuals in the ontology.

When making the aforementioned decisions, the following aspects should be considered: the URIs should be unambiguous, persistent and easily understood and the ontology model should be separated from its instances.

3.3.3 Transform data sets

After having defined the resource naming strategy and selected the ontologies, it is time to transform the data into the RDF format following several steps. First, the RDF serialization has to be selected (RDF/ XML,

[4] See : https://www.w3.org/2011/gld/wiki/Linked_Data_Cookbook#Step_3_Reuse_Vocabularies_Whenever_Possible

Turtle, N-Triples, JSON-LD, etc.). The second step concerns the selection of appropriate tools for data transformation, depending on the format of the data (IFC, spreadsheet, XML, etc.), as the heterogeneity of data formats will require the use of several different tools. The third step is to use the selected tool in order to obtain RDF data. During the conversion process, a mapping between data and the selected ontologies, as well as naming of all instances according to the defined resource naming strategy, will take place. The level of automation of these tasks depends on the conversion tools that will be selected. Furthermore, conversion tools offer possibilities of clearing and structuring messy data in the desired way. An essential aspect of this step concerns the structure and expression of the data in such a way that the desired literals matching can be achieved. This means that not only common instances must exist between the two datasets, but also that they should be expressed in identical lexical forms, which are encoded as Unicode strings, and data types in order to be considered equal (Emmons, Collier, Garlapati, & Dean, 2011). For that reason, based on the data needs of the user, appropriate instances for matching should be identified within each generated dataset. After the transformation is complete, the obtained RDF data set should be evaluated. The evaluation process could include validation of the syntax of the RDF produced as well as execution of SPARQL queries to check the correctness of the obtained results.

3.4 Store the RDF data

The generated RDF data have to be stored into a persistent repository where they can be accessed and queried. A usual choice is to use an RDF triple store, such as a graph-oriented repository that offers the possibility of querying the RDF dataset by using SPARQL. However, there can be several other ways to store RDF data that integrate with the existing infrastructures or architectures of the organization.

3.5. Query development

This task concerns the development of SPARQL queries and its purpose is threefold. First, it serves as a validation mechanism for checking if the produced RDF data express correctly and entirely the datasets that have been used. In case of problematic query results, the data conversion part has to be examined anew. Second, it provides fine-grained querying and reasoning methods on the datasets, meaning that specific pieces of data can be requested, based on certain restrictions and constraints that are input. Third, SPARQL queries, provide real-time views on RDF data stemming from different sources (Michelfeit, 2013). This way, complicated and well-targeted queries could provide timely and precise information by combining data that reside in diverse information silos and were initially published in diverse formats. The query development task includes two main steps: the selection of the queries topics and the actual development of the queries.

3.5.1 Select query topics

Before starting developing SPARQL queries it is essential that the information demands of the organization are clear. A clear concept should exist and the topics of the queries should be selected in such a way that they provide exact data that cover daily data need scenarios of the organization. A typical example of such a concept could be the cost estimation of a construction. Relative query topics would then include requests about the prices of all building components and materials.

3.5.2 Develop SPARQL

After the topics have been specified, it is time to develop/express them into SPARQL queries. Triple patterns, including both resources and variables, are being matched against the available RDF data triples and information is extracted among direct or indirect resources relationships. By using restrictions and constraints on the solutions of the patterns, the desired matching of literals can be performed and specifically-targeted information can be obtained. In order to implement this task, sufficient knowledge and understanding of SPARQL, as well as of the ontologies that have been used are required, in order to formulate the patterns that are to be matched against the RDF triples of the datasets.

4. Proof of concept

In this section, the suggested framework will be tested in the context of simple but representative scenarios of an engineering company's everyday data management practices, where information extracted from a BIM model has to be combined with a differently structured dataset in order to extract data required for specific construction management tasks.

A typical example is when a BIM modeler needs information about door types, their availability and their price at a specific time to choose the most appropriate ones. The products provided by a supplier may be changing,

evolving and even outdated. To avoid rework, when a designed door type is not available, alternative options could be examined when manufacturer's data is linked to the building model.

These illustrative linking tasks will be described below. To implement this example, this study makes use of an existing standard Revit model and a generated dummy set of manufacturer's door data in Microsoft Excel.

4.1 Data collection

The data that will be used in this use case include an architectural BIM model of a two-story building in Autodesk Revit (**Fig. 4**), and a doors catalog in Microsoft Excel (**Fig. 5**).

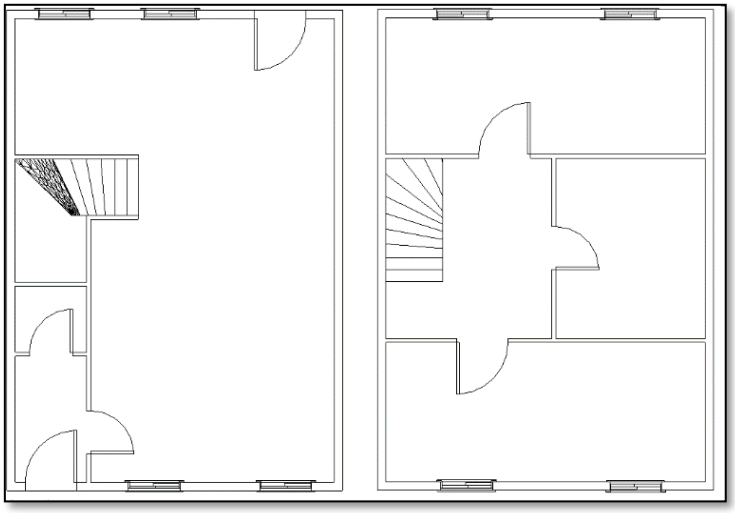

Fig. 4. Floor plans of the Revit model

Dimitris Doors		Enschede, Netherlands Tel: 06 98 65 22 34 email: dimitrisdoors@gmail.com		
Name	Code	Width X Height (mm)	Price (€)	Stock
D3454	EN658075	0762 X 2032	130	15
D3455	EN658071	0915 X 2134	145	14
D3456	EN658067	0813 X 2134	150	0
D3457	EN658063	0864 X 2032	165	2
D3458	EN658059	0864 X 2134	175	3
D3459	EN658055	0915 X 2032	180	5
D3460	EN658051	0864 X 2032	170	8
D3461	EN658047	0864 X 2134	175	18
D3462	EN658085	0762 X 2134	160	0
D3463	EN658086	0864 X 2032	160	24
D3464	EN658080	0915 X 2032	175	32
D3465	EN658076	0762 X 2032	155	0
D3466	EN658089	0762 X 2134	185	10
D3467	EN658092	0864 X 2134	190	0

Fig. 5. Dummy Excel spreadsheet representing door manufacturer's object data

4.2 Data analysis

Regarding the architectural Revit model, a two-story building including architectural walls, floors, doors, windows, and a staircase was exported to IFC data model, which is based on the EXPRESS schema. The generated file is expressed in the IFC-SPF format ('STEP file'), which is a text format that defines the entities of the model, as well as the way that they relate to each other.

As far as the manufacturer's data are concerned, they include a list of all the doors that they produce, including their product names, according to the naming system of the company, their product codes, their dimensions, their unit prices, and their inventory level. The data are structured in a tabular form in the Excel format.

4.3. Data conversion

4.3.1 Select ontology

In this step, ontologies that describe the terms, data, and relationships within the two datasets were selected.

Regarding the IFC model, the ifcOWL[5] ontology was selected. IfcOWL has been jointly standardized under the umbrella of buildingSMART (buildingSMART, 2016) in order to cover the need for formalization and convergence of the effort to convert the EXPRESS based schema IFC to an OWL ontology, which had been researched for several years (Schevers & Drogemuller, 2006; Beetz et al., 2009; Gao et al., 2015; Pauwels & Deursen, 2015) and serves as a domain ontology for the construction industry. IfcOWL constitutes a translation of modeling constructs of the IFC model defined in the EXPRESS language (ENTITY, Attributes, data types, etc.) into equivalents from the RDF(S) and OWL modeling vocabularies (Class, ObjectProperty, DatatypeProperty, domains, ranges, etc.) resulting in an OWL meta-model for buildings. According to this schema mapping, instance models exported in IFC can be represented as RDF data without loss of information (Pauwels et al., 2018). Therefore, for the purpose of the current case, ifcOWL was deemed appropriate to describe the geometric attributes of the building's doors, which were used for the literals matching task.

In order to semantically describe the manufacturer's door data, the GoodRelations ontology[6] (Hepp, 2008) was selected. This ontology was created in order to cover representational business scenarios and can be used to exchange information about products and services, pricing, payment options, other terms and conditions, store locations and their opening hours, and many other aspects of e-commerce. GoodRelations ontology can be used for Product Information Management inside an organization or a value chain as a global database schema for integrating information about products and services from multiple sources. Although this ontology was not developed specifically for the construction domain, and therefore lacks AEC-FM industry-specific vocabulary, it was deemed appropriate for the current use case, since the data that needed to be semantically described concern dimensions, product codes, prices, and inventory levels, which are typical aspects of any domain products, and GoodRelations included sufficient vocabulary for their description.

4.3.2 Define the resource naming strategy

The URIs for this research were constructed based on tips provided by Overbeek & van den Brink (2013) and Sauermann & Cyganiak (2008), who suggest that URIs should have the following composition:

```
http://{domain}/{type}/{concept}/{reference},
```

where the `{domain}` component contains the Internet domain and optionally a path within this domain, the `{type}` component indicates which kind of URI is involved and may be `'id'` (identifier of an object in a register) `'doc'` (documentation on the object in the register) or `'def'` (definition of a term in an ontology), the `{concept}` gives an indication of the type of the concept that is identified by the URI and the `{reference}` is the identifying name or code of the individual object.

For both datasets, it was chosen to use http://example.com/[7] as a base URI because this domain is free to use without prior coordination or asking for permission. Then, according to the aforementioned pattern, the URI for the IFC data is:

```
http://example.com/id/ifc/20190608_140143/,
```

where the last component is automatically assigned to the URI by the conversion tool that was used (see 5.3.3) and acts as an identifier of the specific IFC file that was converted and includes the date of conversion, followed by a numeric code.

Regarding the manufacturer's data, the respective URI of each door element is as follows:

```
http://example.com/id/door/D1234/,
```

[5] See : http://github.com/buildingSMART/ifcOWL
[6] See : http://www.heppnetz.de/ontologies/goodrelations/v1
[7] See: http://example.com/

where the last component is generated on the conversion tool that was used (see 5.3.3) and is named after the product name of each door, namely the first column of the catalog table.

4.3.3 Transform data sources

Due to the heterogeneity of the data formats, different tools have been selected for the transformation process. For the IFC data, the IFCtoRDF[8] converter was selected. This tool, which is a set of reusable Java components, converts IFC STEP formatted BIM files into RDF graphs, conforming to the ifcOWL ontology, in a straightforward way. Using the provided desktop user interface (**Fig. 6**), a file to file conversion was performed using the Turtle serialization syntax. A part of the output Turtle file, including data related to an IfcDoor, is provided in Appendix A.

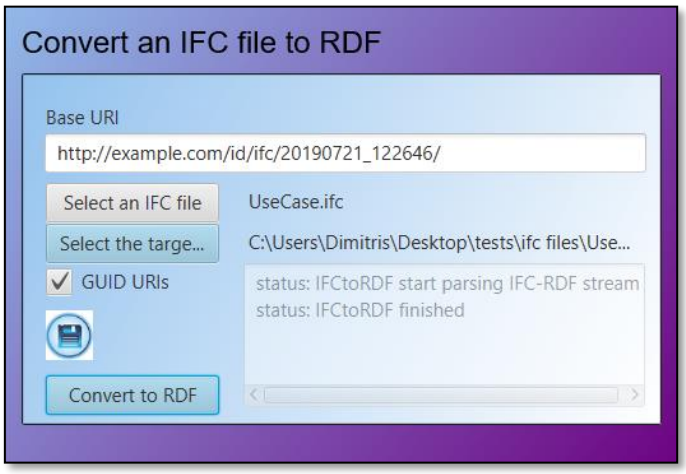

Fig. 6. Screenshot of IFCtoRDF converter's Graphical User Interface

For the Manufacturer's data, Open Refine[9] was used. This is a powerful tool for working with messy data that offers possibilities of cleaning it, transforming it from one format into another, and extending it with web services and external data. Its RDF extension enables the mapping of the values in the columns of the spreadsheet into RDF triple formats. In the current use case, a restructuring of the data was necessary. More specifically, the dimensions of the doors had to be split into two separate columns, namely weight and height, so that each dimension has its own value. Furthermore, it had to be made sure that the dimensions of the doors are expressed in the same units and the same data type as in the IFC model so that the matching would be feasible. The generated RDF dataset, serialized in Turtle, is available in Appendix B.

4.4 Store the RDF data

It was decided to store the generated RDF datasets into GraphDB[10]. It is a highly-efficient and robust graph database that offers the possibility of storing RDF datasets in local or remote repositories, exploring the datasets and querying them due to its SPARQL support. For demonstration purposes, the manufacturer's data were stored into a separate local repository, in order to demonstrate the function of federated queries. These enable access to diverse repositories and the querying of data stemming from different sources. Therefore, the case where the manufacturer already provides his data in the RDF format is also included.

4.5. Query development

4.5.1 Select query topics

The query topics have been selected in such a way that they satisfy the threefold purpose that has been described in 4.5 and include requests that provide fine-grained views on datasets as well as complicated requests that require the integration of the two datasets.

[8] See: http://github.com/jyrkioraskari/IFCtoRDF-Desktop
[9] See: http://openrefine.org/
[10] See: http://graphdb.ontotext.com/documentation/free/index.html

For the chosen case, two data need scenarios will be displayed alongside the respective queries that are required for each scenario. In the first one, we suppose that the BIM modeler uses the door catalog for the first time and therefore needs to query the catalog in order to select the appropriate door types.

In the second scenario, we suppose that the modeler has already chosen door types from the catalog and now wants to confirm the availability of these. In case of no availability, alternative options would have to be sought.

4.5.2 Develop SPARQL

In this section, the SPARQL queries that have been developed in order to provide the requested data, according to each scenario, will be presented alongside a short description of their functionality and their results. A concentrated presentation of all the queries that were developed is available in Appendix C.

4.5.2.1 Scenario 1

In this scenario, the user wishes to select the appropriate door types. In order to do that, first, the amount and size of doors that are needed would have to be extracted from the IFC dataset. This task is performed with the following Query 1, where the patterns that will be matched against the RDF triples within the dataset constitute the main part of the SPARQL query. In order to do that, sufficient knowledge of the way in which the data are described by the ontology is required. Statements such as "doorD is an IfcDoor", "doorD has overallHeight h" etc. enable the navigation within the RDF dataset of the IFC data, which describe the whole building, and the identification and retrieval of the specific door-related data that we are interested in. Although this query does not concern the integration between BIM and the catalog data, it displays how SPARQL can provide access to specific pieces of data and let the user have an overview of the data they are interested in.

```
SELECT Distinct  ?doorD ?IfcIdentifier ?hD ?wD

WHERE {
  ?doorD a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifcowl:tag_IfcElement ?tag.
  ?tag express:hasString ?IfcIdentifier.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
}
```

Fig. 6. Query 1

Table 1. Query 1 results

doorD	IfcIdentifier	hD	wD
inst:GUID/145adf16-8b67-460a-bca8-932e5c983647	330805	2032.	915.
inst:GUID/145adf16-8b67-460a-bca8-932e5c983685	330999	2134.	762.
inst:GUID/145adf16-8b67-460a-bca8-932e5c9832bd	329935	2032.	762.
inst:GUID/145adf16-8b67-460a-bca8-932e5c98325c	329774	2032.	762.
inst:GUID/145adf16-8b67-460a-bca8-932e5c983de7	329621	2134.	915.
inst:GUID/145adf16-8b67-460a-bca8-932e5c98318e	330748	2032.	915.
inst:GUID/145adf16-8b67-460a-bca8-932e5c98332e	330076	2134.	915.

This query scans the whole model and isolates specific information related to the doors without the need to access the Revit model itself. In **Table 1**, The URI and the IFC Identifiers of each of the model's seven doors are provided alongside their dimensions so that the user can look them up in the model in case they need to.

Next logical step would be to query the door catalog in order to look for appropriate door types for each of the model's doors. Query 2 below shows this process. For every pair of dimensions, the number of doors that are needed in the model, as well as the corresponding door types in the catalog, with their inventory level and their unit prices are identified. In order to do that, the height and width of the doors have to be matched between the

two datasets. This literals matching of the attributes height and width enables the interaction between the two RDF datasets and provides integration of data residing in diverse sources. The matching constraints are highlighted in **Fig. 7** below.

```
SELECT distinct ?hS ?wS (COUNT(*)/2 AS ?DoorsNeeded) ?doortypecode
?DoorsAvailable ?UnitPrice

WHERE {
  ?doorD a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifc:name_IfcRoot ?root.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
  ?panel ifc:name_IfcRoot ?root.
  ?doorStyle ifc:hasPropertySets_IfcTypeObject ?panel.
  ?typedef ifcowl:relatingType_IfcRelDefinesByType ?doorstyle;
           ifcowl:relatedObjects_IfcRelDefines ?doorD;
           ifcowl:relatedObjects_IfcRelDefines ?IfcDoor.
Service <http://10.99.99.4:7200/repositories/exceldata> {
?doorS gr:serialNumber ?doortypecode;
    gr:width ?wS;
    gr:height ?hS;
    gr:hasPriceSpecification ?UnitPrice;
    gr:hasInventoryLevel ?Inventory.
    }
Filter (?wD = ?wS && ?hS = ?hD).
Bind (?Inventory AS ?DoorsAvailable).
}
GROUP BY ?wS ?hS ?DoorsAvailable ?doortypecode ?UnitPrice
```

Fig. 7. Query 2

Table 2. Query 2 results

hS	wS	DoorsNeeded	doortypecode	DoorsAvailable	UnitPrice
2134	762	1	EN658085	0	160
2134	762	1	EN658089	10	185
2032	762	2	EN658075	15	130
2032	762	2	EN658076	0	155
2134	915	2	EN658071	14	145
2032	915	2	EN658055	5	180
2032	915	2	EN658080	32	175

This query introduces a very powerful concept in SPARQL, the federated query. A federated query is performed in Query 2 in order to connect with the manufacturer's data, which is stored in a separate repository and retrieve data included in the doors catalog. This functionality is achieved with the introduction of the `SERVICE` statement, followed by the repository's URL. Regarding doors with dimensions 2134x762mm, the result of this query shows that there is one door needed in the model with these dimensions, while there are 2 different door types in the door catalog matching these dimensions, namely the ones with door type code EN658085 and EN658089 respectively. It can also be seen that regarding the first door type there no doors available in the manufacturer's warehouse, while for the second one there are ten doors currently available. Therefore, the options of the modeler regarding the types of doors with dimensions 2134x762mm are limited to one, namely the door type with code EN658089, which costs 185 euros per unit. In the same way, the available options for all doors can be identified.

4.5.2.2 Scenario 2

In this scenario, we consider the case where the modeler has already preselected specific door types for each of the doors in the model. Therefore, there is a need to confirm that these door types are available at any time, according to the updated catalog that is provided by the manufacturer. In case there are no doors available, alternative options would have to be sought. The door type that has been selected for each door has been input in the model as an IFC parameter, as can be seen in **Fig. 8** below.

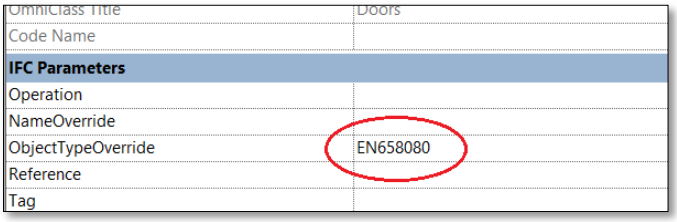

Fig. 8. Door type code input in the model, as an IFC parameter

The existence of the manufacturer's door type code in the BIM model, besides the manufacturer's door data Excel spreadsheet, gives more opportunities for literals matching and therefore for the execution of more targeted queries. Query 3 below asks which door types have been selected by the modeler, how many doors have been assigned to each door type and the inventory level of each door type. Again, the constraints that define the matching of literals, and therefore the interaction of the two datasets, is highlighted. According to these, in order to get the desired data, the height, width and door type codes should match between the two datasets

```
SELECT distinct ?doortypecode  (COUNT(*)/2 AS ?DoorsNeeded) ?DoorsAvailable
?UnitPrice
WHERE {
  ?doorD a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifc:name_IfcRoot ?root.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
  ?panel ifc:name_IfcRoot ?root.
  ?doorStyle ifc:hasPropertySets_IfcTypeObject ?panel;
             ifc:name_IfcRoot [express:hasString ?inputcode].
  ?typedef ifcowl:relatingType_IfcRelDefinesByType ?doorstyle;
           ifcowl:relatedObjects_IfcRelDefines ?doorD;
           ifcowl:relatedObjects_IfcRelDefines ?IfcDoor.
Service <http://10.99.99.4:7200/repositories/exceldata> {
?doorS a gr:ProductOrService;
    gr:name ?name;
    gr:serialNumber ?doortypecode;
    gr:width ?wS;
    gr:height ?hS;
    gr:hasPriceSpecification ?UnitPrice;
    gr:hasInventoryLevel ?Inventory.
    }
filter (?wS = ?wD && ?hS = ?hD && ?doortypecode = ?inputcode).
Bind (?doorD AS ?DoorsNeeded).
Bind (?Inventory AS ?DoorsAvailable).
}
GROUP BY ?doortypecode  ?DoorsAvailable ?UnitPrice
```

Fig. 9. Query 3

Table 3. Query 3 results

doortypecode	DoorsNeeded	DoorsAvailable	UnitPrice
EN658075	2	15	130
EN658071	2	14	145
EN658080	2	32	175
EN658085	1	0	160

A quick overview of the results shows that four different door types have been selected. There are 2 doors needed for every of the following door types codes, namely EN658075, EN658071, and EN658080, while the available items based on the manufacturer's list are 15, 14 and 32 respectively. Therefore, there are no availability issues. However, concerning the door type code EN658085, there is 1 door needed, while 0 doors are available at the moment by the manufacturer. Therefore, a replacement should be issued and the next logical step would be to check for alternative options. In order to do so, for every Ifc door, we should search in the manufacturer's dataset for door types that have the same dimensions with it, have not been already selected and their inventory level is higher than zero. This task is performed in Query 4 below.

```
SELECT Distinct  ?reldoor ?hD ?wD ?CurrentType ?AltType ?amount

WHERE {
  ?door a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifc:name_IfcRoot ?root.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
  ?panel ifc:name_IfcRoot ?root.
  ?doorStyle ifc:hasPropertySets_IfcTypeObject ?panel;
             ifc:name_IfcRoot [express:hasString ?inputcode].
  ?typedef ifcowl:relatingType_IfcRelDefinesByType ?doorstyle;
           ifcowl:relatedObjects_IfcRelDefines ?door;
           ifcowl:relatedObjects_IfcRelDefines ?reldoor.

Service <http://10.99.99.4:7200/repositories/exceldata> {
?doorS a gr:ProductOrService;
    gr:name ?name;
    gr:serialNumber ?doortypecode;
    gr:width ?wS;
    gr:height ?hS;
    gr:hasPriceSpecification ?price;
    gr:hasInventoryLevel ?amount.
    }
filter (?wS = ?wD && ?hS = ?hD && ?inputcode != ?doortypecode && ?amount != 0.0 ).
Bind (?inputcode AS ?CurrentType).
Bind (?doortypecode AS ?AltType).
 }
```

Fig. 10. Query 4

Table 4. Query 4 results

reldoor	hD	wD	CurrentType	AltType	amount
inst:GUID/145adf16-8b67-460a-bca8-932e5c983685	2134.	762.	EN658085	EN658089	10
inst:GUID/145adf16-8b67-460a-bca8-932e5c983647	2032.	915.	EN658080	EN658055	5
inst:GUID/145adf16-8b67-460a-bca8-932e5c98318e	2032.	915.	EN658080	EN658055	5

According to the results of this query, the door of type EN658085 can be replaced with one of type EN658089, as there are ten doors of this type in the manufacturer's warehouse. Moreover, available alternative options are provided for the rest of the door types as well, in case the user wishes to replace any other door types.

5. Evaluation - Discussion

In this paper, an approach for integrating BIM with relevant cross-domain data, based on RDF, SPARQL and literals matching was explored. A conceptual framework of the approach was presented and tested in the context of a typical scenario, where an engineering company needs to combine data from a BIM model with relevant heterogeneous data. In this section, the results of the study are discussed and the tradeoffs, the potential, the limitations and the implications of the approach are addressed.

Overall, practically speaking, in the context of the use case and the data that were used, the proposed framework was proven quite functional, in terms of providing possibilities of semi-automating the integration of BIM with data residing at diverse, heterogeneous sources. By following the consecutive steps of the framework, and having a Revit model and an Excel spreadsheet containing door manufacturer's data as a starting point, it was possible to successfully combine data from both sources and semi-automate tasks that would otherwise require manual extraction and combination of these by assessing whole documents, which could be a rather laborious and error-prone task. Such tasks include aggregating and listing the door types designed in the model based on their dimensions, identifying the suitable available door types in the manufacturer's catalog as well as the alternative options in case a replacement is needed.

One basic assumption of the current study was that the explored approach has the potential to achieve data integration in a more comprehensive and familiar to the user way and, in general terms, this was the case. To begin with, the data conversion process was quite smooth due to the use of available tools that facilitated the process. The difference of this approach compared to related work lies in the fact that existing ontologies were selected to semantically describe the datasets, instead of developing new ones. Regarding the datasets that were used for the proof of concept, the search and the selection of suitable ontologies was quite easy, as the popular and widely used ifcOWL and GoodRelations were deemed sufficient for the description of the two datasets. Furthermore, it was proven that by providing an alternative way of interacting between different datasets, namely by matching literals, such as product codes, height and width values, at query level, integration of diverse data is possible while avoiding the establishment of semantic links between the respective ontologies. This way, the complicated tasks of ontology development and alignment, which require significant amounts of understanding, reflection and labor were circumvented, while the practical purpose of the approach, namely the integration of heterogeneous data was fulfilled. By applying constraints within the SPARQL queries, that require the matching of specific literals, such as height, width and product code, using the Filter construct, it was possible to relate data within the manufacturer's catalog to data within the BIM model in a rather practical way and extract useful information about the types, the availability and the price of doors. The basic aspect that renders the approach of literals matching more familiar to the user is the fact that it actually automates the otherwise manual process of combining the 3D model with the manufacturer's door data per se. This task normally requires access to both the Revit model and the Excel file, manual matching of the doors designed in the model with the door types provided by the manufacturer, and assessment of their suitability, availability and price. These exact tasks were automated due to the explored approach, while the rationale of the data combination process remained the same, maintaining the complexity of Semantic Web adoption at a lower level. Therefore, after comprehending the basic aspects of RDF, OWL and SPARQL, this approach could be easier adopted by practitioners, compared to ontology alignment that requires a lot of reflection and experience.

However, the implementation of the conceptual approach revealed that in order to achieve the desired functionality, there are several requirements that have to be met. First, in order to avoid the development of new ontologies, it is essential to find ones that sufficiently describe the datasets or at least the parts of these that will

be used in the integration process. Regarding the BIM data, the selected IfcOWL is an officially standardized vocabulary for describing modeling constructs by performing an exact syntax mapping between the EXPRESS language and OWL. Furthermore, inspired by the IFC, several modular ontologies such as the Building Topology Ontology (BOT), Product Ontology (PRODUCT), Properties Ontology (PROPS) and Geometry Ontology (GEOM) have been developed in order to capture the essence of most of the IFC data sets, namely geometry, building topology, products, and their properties (Pauwels et al., 2018), enabling thus their use in a wide range of use cases and applications. As far as the manufacturer's data are concerned, the GoodRelations ontology includes appropriate properties that were able to describe the product code, the dimensions, the price and the inventory level of the manufacturer's doors. Nonetheless, this ontology has not been developed specifically for construction products. It lacks a definition of basic properties needed for their sufficient description, such as material properties, but provides guidelines to extend it (Radinger, Rodriguez-castro, Stolz, & Hepp, 2013). Given the fact that the nature, as well as the structure of the external relevant data that an engineering company wishes to integrate with their BIM models, can widely vary, it is unlikely that existing ontologies can always describe the datasets in question accurately and fully. Therefore, development of new ontologies or at least extension of existing ones seem inevitable.

Another basic prerequisite for the successful implementation of the approach is the lexical and data type similarity of the literals that are to be matched. That means that typographic mistakes, expression of properties using different units (e.g. a length could be expressed in meters instead of millimeters) or representation of these in different data types (e.g. a date can be represented in the format of "string", instead of "Date" or "DateTime") could prohibit the desired matching between the literals. The larger and more complicated the datasets that are to be processed, the higher the possibility of the occurrence of such discrepancies between them, and therefore of the failure of the literals matching process.

Furthermore, the fact that matches of specific literals have to be identified between the datasets and defined as Filter constraints in order to achieve the desired functionality of the approach presupposes that in order to construct efficient SPARQL queries, thorough prior knowledge of the data is required. In the case of large and complex datasets, applying these matching constraints would be rather time-consuming and error-prone and would lead to significantly more complicated queries.

At the same time, although ontology development and alignment are perceived complicated and their deployment requires significant amounts of effort, once implemented correctly they eliminate the aforementioned issues. More specifically, a well-constructed ontology can constitute a rich vocabulary that can capture the whole range of a domain of action of an organization and can offer plenty of possibilities of links to other ontologies. These links can be performed either at instance-level, namely at the level of individual entities (e.g. by denoting that two instances in different ontologies actually refer to the same thing) or at schema-level to declare relationships between two properties or two classes from different vocabularies. (e.g. by denoting that one instance is a subclass of another) (Acosta et al., 2019). By establishing such links it is possible to relate/link instances between two ontologies regardless of their literal values. For example, it can be stated that IfcDoor_1867, as defined in IfcOWL, is the same as Door6587 as defined in a manufacturer's ontology, or that the manufacturer's door types with specific dimensions are subclasses of the equivalent IfcDoors having the same dimensions. This way, implicit knowledge that was applied at the query level in the tested approach, thus increasing significantly the complexity of the queries, can become explicit by being applied at an ontological level.

By highlighting the challenges that arise when attempting to circumvent ontology development and alignment, the results of this study come to corroborate the crucial role of ontology linking in achieving integration of BIM with cross-domain data, in order to enhance data management practices in the construction sector and address the wide heterogeneity of data that have to be assessed, interpreted and combined. The outcome of this study constitutes an input for engineering companies in their effort to comprehend the requirements, the potential and the challenges of deploying Semantic Web technologies within their everyday data management practices.

Although the discussion and evaluation of the results revealed several limitations of the tested approach, mainly regarding its generalizability, it can still serve as a learning tool which can be deployed in a narrow scope by an engineering company that takes the first step towards Semantic Web technologies. In the context of simple scenarios, this approach could help practitioners automate their construction data management practices and become familiar with the essential processes of semantic data modeling and SPARQL, before deciding to invest in solutions that require the development and semantic linking of ontologies.

Within this research, two main limitations can be identified. First, the data that were used were dummy data created specifically for the purposes of the current research and the domain, the nature and the structure of the data were selected to fit the needs of the study. That option underlies the risk of selection bias and could have been eliminated by using actual datasets.

Second, the conclusions regarding the feasibility, comprehensibility and generalizability of the explored approach, were drawn based on the researcher and his own experience with the process, which could also lead to biased judgment. The conduction of a workshop with the participation of employees of an engineering company, could lead to more valid and reliable conclusions. However, due to time constraints, this option was not possible.

6. Conclusions

This paper explored the potential of implementing Semantic Web technologies in a setting where an engineering company wished to integrate BIM with relevant data by using existing ontologies and avoiding the establishment of semantic links between them. The hypothesis that was tested was that such an approach would limit the complexity of the data integration process and would, therefore, be easier to apply.

The motivation for this study was the observation of academics that ontology development and alignment tasks require reflection and hard work from domain experts and ontology engineers and at the same time-related researches do not document or sufficiently explain linking decisions, which require human interpretation. This combined with the rather limited knowledge of the industry regarding Semantic Web technology, slow down its adoption. For that reason, a conceptual framework of an RDF-based integration approach depending on literals matching was presented and tested in the context of a scenario where a BIM model had to be combined with an Excel file including manufacturer's data about doors.

The results of the evaluation showed that although proven successful for linking datasets, the unlikelihood of the existence of ontologies that properly describe the datasets and cover the data needs of every company, the required restrictions regarding the lexical and data type similarity of the literals, the required prior specific knowledge about the data and the resulting complexity of the queries, render this approach hard to be properly applied and generalized. In the context of everyday construction management tasks of an engineering company, the need to assess numerous files of various types and sizes and combine them with BIM would require an integration method that can be effectively applied regardless of the size and type of the datasets. The amount of work needed and generalizability are basic criteria regarding the effectiveness of a data integration approach and the evaluation results reveal that the tested approach poses limitations to both.

By highlighting and reflecting on the limitations of this alternative approach, the results of this study assist engineering companies with limited knowledge regarding Semantic Web technologies to comprehend their underlying challenges as well as the benefits that they can bring to their everyday data management practices. More specifically it became clear that the purpose of deploying these technologies should be to limit the commitments, the restrictions and the amount of work needed every time, regarding the data integration process. It was revealed that seemingly simpler and easier solutions might involve numerous requirements and case-specificity, which renders them complex and not easily generalized. The importance of using ontologies that fully describe the datasets in question was discussed, alongside the difficulty of always finding available ontologies that cover the user's needs. The need for developing new or at least extending existing ones was also highlighted. Moreover, the value of ontologies for making implicit knowledge explicit was elaborated. Defining relationship statements among instances or classes by establishing semantic links between ontologies, instead of doing it implicitly at the query level, results in less complicated, less specific and more comprehensible queries.

Given the issues raised regarding the complexity of ontology development and alignment and the lack of their explanation by academics, more detailed documentation is recommended. Since these tasks are subject to human interpretation, the considerations and measures that need to be taken regarding the development of ontologies as well as the creation of links, should be elaborated and research accomplishments should be accessible in order to facilitate comprehension and easier reuse, in an industry that generally lacks knowledge and experience on Semantic Web technologies.

As far as practical recommendations are concerned, engineering companies should realize that given the fact that these technologies are so different than traditional development practices, a significant amount of effort has to be made, at least at the early stages. Gradual adoption could be achieved via training and after the knowledge has been acquired, time, deliberation and reflection would be required in order to establish practices of semantic data modeling and integration and enhance their everyday construction management tasks.

Acknowledgements

I would like to thank the University of Twente and specifically Prof. Dr. Ir. A.G. Dorée, Dr. Ir. L.L. olde Scholtenhuis and Dr. Ir. F. Vahdatikhaki for their support, their feedback and for sharing their knowledge with me. I would like to thank M. Veerman for his support and for giving me the opportunity to perform my master thesis research at Witteveen+Bos.

7. Appendixes

Appendix A. Partial RDF file of IFC data

Appendix B. RDF file of manufacturer data

Appendix C. Proof of concept queries

8. References

Acosta, B., Fagnoni, E., Norton, E., Maleshkova, M., Doningue, M., Mikroyannidis, J., & Mulholland, A. (2019). *How to use Linked Data*.

Beetz, J., Leeuwen, J. O. S. V. A. N., & Vries, B. D. E. (2009). IfcOWL : A case of transforming EXPRESS schemas into ontologies, 89–101. https://doi.org/10.1017/S0890060409000122

Berners-Lee, T. (1998). The World Wide Web: A very short personal history. Retrieved from https://www.w3.org/People/Berners-Lee/ShortHistory.html

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

Berners Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web: A new form of Web content that is meaningful to computers will unleash the revolution of new possibilities. *The Scientific American*, *5*(284), 34–43. https://doi.org/10.1038/scientificamerican0501-34

buildingSMART. (2016). ifcOWL ontology (IFC4). Retrieved from http://ifcowl.openbimstandards.org/IFC4/index.html

Cheatham, M., & Pesquita, C. (2017). Semantic Data Integration. In *Handbook of Big Data Technologies*. https://doi.org/10.1007/978-3-319-49340-4_8

Chou, J. S., & Ngo, N. T. (2016). Smart grid data analytics framework for increasing energy savings in residential buildings. *Automation in Construction*, *72*, 247–257. https://doi.org/10.1016/j.autcon.2016.01.002

Costa, G, & Madrazo, L. (2015). Connecting building component catalogues with BIM models using semantic technologies : an application for precast concrete components. *Automation in Construction*, *57*, 239–248. https://doi.org/10.1016/j.autcon.2015.05.007

Costa, Gonçal. (2017). *Integration of building product data with BIM modelling: a semantic-based product catalogue and rule checking system*. https://doi.org/10.13140/RG.2.2.11123.63522

Cuenca, J., Larrinaga, F., & Curry, E. (2017). A unified semantic ontology for energy management applications. *CEUR Workshop Proceedings*, *1936*, 86–97.

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. https://doi.org/10.1016/j.aei.2012.10.003

Emmons, I., Collier, S., Garlapati, M., & Dean, M. (2011). RDF Literal Data Types in Practice, 1–13.

Ferrara, A., & Nikolov, A. (2011). Data Linking for the Semantic Web, *7*(September), 46–76. https://doi.org/10.4018/jswis.2011070103

Gao, G., Liu, Y., Wang, M., Gu, M., & Yong, J. (2015). A query expansion method for retrieving online BIM resources based on Industry Foundation Classes. *Automation in Construction*, *56*, 14–25. https://doi.org/10.1016/j.autcon.2015.04.006

Garshol, L. M. (2013). Semantic Web adoption and the users. Retrieved from http://www.garshol.priv.no/blog/261.html

Godager, B. (2018). Critical review of the integration of bim to semantic web technology. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(4), 303–308. https://doi.org/10.5194/isprs-archives-XLII-4-233-2018

Heath, T., & Bizer, C. (2011). *Linked Data. Evolving the Web into a Global Data Space*. Morgan & Claypool.

Hepp, M. (2008). GoodRelations : An Ontology for Describing Products and Services Offers on the Web, 329–346.

Hitzler, P., Krötzsch, M., Parsia, B., & Sebastian, R. (2012). Web Ontology Language (OWL). Retrieved from https://www.w3.org/OWL/

Hor, A., Jadidi, A., & Sohn, G. (2016). BIM-GIS INTEGRATED GEOSPATIAL INFORMATION MODEL USING SEMANTIC WEB AND RDF GRAPHS, *III*(July), 73–79. https://doi.org/10.5194/isprsannals-III-4-73-2016

Hyland, B., Atemezing, G., & Villazón-Terrazas, B. (2014). Best Practices for Publishing Linked Data. Retrieved from http://www.w3.org/TR/2014/NOTE-ld-bp-20140109/

Kalpoe, R. (2016). *The integration of dispersed building information by using Linked Data principles*.

Kuriakose, J. (2009). Understanding and Adopting Semantic Web Technology. *Cutter IT Journal, Vol. 22(9)*.

Lanthaler, M., & Gütl, C. (2011). A Semantic Description Language for RESTful Data Services to Combat

Semaphobia, *5*(June), 47–53.
Lee, S., Kim, K., & Yu, J. (2014). BIM and ontology-based approach for building cost estimation. *Automation in Construction*, *41*, 96–105. https://doi.org/10.1016/j.autcon.2013.10.020
Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Richter, S., Chipman, T., … Wix, J. (2013). IFC4 Official Release. Retrieved November 29, 2018, from http://www.buildingsmart-tech.org/ifc/IFC4/final/html/index.htm
Michelfeit, J. (2013). *Linked Data Integration*. Charles University in Prague.
Niknam, M. (2017). Supply Chain Semantic Information Modeling for Data Integration with BIM and FM, *3*(4), 3–4. https://doi.org/10.15406/mojce.2017.03.00077
Niknam, M., & Karshenas, S. (2017). A shared ontology approach to semantic representation of BIM data. *Automation in Construction*, *80*, 22–36. https://doi.org/10.1016/j.autcon.2017.03.013
Overbeek, H., & van den Brink, L. (2013). Towards a national URI-Strategy for Linked Data of the Dutch public sector, (2), 1–19.
Parsanezhad, P., & Dimyadi, J. (2014). Effective Facility Management and Operations via a BIM-based Integrated Information System, 1–12.
Pauwels, P., & Deursen, D. Van. (2015). IFC / RDF : Adaptation , Aggregation and Enrichment, (March 2012).
Pauwels, P., Krijnen, T., Terkaj, W., & Beetz, J. (2017). Enhancing the ifcOWL ontology with an alternative representation for geometric data. *Automation in Construction*, *80*, 77–94. https://doi.org/10.1016/j.autcon.2017.03.001
Pauwels, P., McGlinn, K., Törmä, S., & Beetz, J. (2018). Linked Data. In *A. Borrmann, M. König, C. Koch, & J. Beetz (Eds.), Building Information Modeling* (pp. 181–197). Cham: Springer International Publishing.
Pauwels, P., Zhang, S., & Lee, Y. C. (2017). Semantic web technologies in AEC industry: A literature overview. *Automation in Construction*, *73*, 145–165. https://doi.org/10.1016/j.autcon.2016.10.003
Petkova, T. (2019). Data, Databases and Deeds: A SPARQL Query to the Rescue. Retrieved from https://www.ontotext.com/data-databases-sparql-query/
Prud'hommeaux, E., & Buil-Aranda, C. (2013). SPARQL 1.1 Federated Query. Retrieved from https://www.w3.org/TR/sparql11-federated-query/
Radinger, A., Rodriguez-castro, B., Stolz, A., & Hepp, M. (2013). BauDataWeb : The Austrian Building and Construction Materials Market as Linked Data.
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. https://doi.org/10.1016/j.autcon.2015.04.002
Sauermann, L., & Cyganiak, R. (2008). Cool URIs for the Semantic Web. Retrieved July 6, 2019, from https://www.w3.org/TR/cooluris/
Schevers, H., & Drogemuller, R. (2006). Converting the Industry Foundation Classes to the Web Ontology Language, (Skg 2005), 2005–2007.
Schreiber, G., & Raimond, Y. (2014). RDF 1.1 Primer. Retrieved January 5, 2019, from https://www.w3.org/TR/rdf11-primer/
Serrano, D., & Stroulia, E. (2017). Linked REST APIs : A Middleware for Semantic REST API Integration, 138–145. https://doi.org/10.1109/ICWS.2017.26
Soibelman, L., Wu, J., Caldas, C., Brilakis, I., & Lin, K. Y. (2008). Management and analysis of unstructured construction data types. *Advanced Engineering Informatics*, *22*(1), 15–27. https://doi.org/10.1016/j.aei.2007.08.011
Vandenbussche, P., Atemezing, G. A., & Vatant, B. (2014). Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web, *1*, 1–5.
Wood, D., Zaidman, M., Ruth, L., & Hausenblas, M. (2014). *Linked Data: Structured data on the Web*. Shelter Island, NY: Manning Publications Co.
Yu, M. (2016). *A linked data approach for information integration between BIM and sensor data*. Retrieved from https://pure.tue.nl/ws/portalfiles/portal/61132956/Yu_0925649.pdf

Appendix A. Partial RDF file of IFC data

```
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#> .
@prefix inst:   <http://example.com/id/ifc/20190721_122646/> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix express:<https://w3id.org/express#> .
@prefix list:   <https://w3id.org/list#> .

inst:   rdf:type     owl:Ontology ;
        owl:imports  ifcowl: .

<http://example.com/id/ifc/20190721_122646/GUID/145adf16-8b67-460a-bca8-
932e5c98325c>
        a                              ifcowl:IfcDoor ;
        ifcowl:globalId_IfcRoot        inst:IfcGloballyUniqueId_17832 ;
        ifcowl:name_IfcRoot            inst:IfcLabel_17819 ;
        ifcowl:objectPlacement_IfcProduct
                inst:IfcLocalPlacement_11937 ;
        ifcowl:objectType_IfcObject    inst:IfcLabel_17777 ;
        ifcowl:overallHeight_IfcDoor   inst:IfcPositiveLengthMeasure_17817 ;
        ifcowl:overallWidth_IfcDoor    inst:IfcPositiveLengthMeasure_17812 ;
        ifcowl:ownerHistory_IfcRoot    inst:IfcOwnerHistory_41 ;
        ifcowl:representation_IfcProduct
                inst:IfcProductDefinitionShape_10949 ;
        ifcowl:tag_IfcElement          inst:IfcIdentifier_17833 .

inst:IfcGloballyUniqueId_17832
        a                  ifcowl:IfcGloballyUniqueId ;
        express:hasString  "0KMjyMYsT62hoeaovSc39S" .

inst:IfcLabel_17819  a     ifcowl:IfcLabel ;
        express:hasString  "M_Single-Flush:0762 x 2032mm:329774" .

inst:IfcLocalPlacement_11937
        a       ifcowl:IfcLocalPlacement ;
        ifcowl:placementRelTo_IfcLocalPlacement
                inst:IfcLocalPlacement_11925 ;
        ifcowl:relativePlacement_IfcLocalPlacement
                inst:IfcAxis2Placement3D_11936 .

inst:IfcLabel_17777  a     ifcowl:IfcLabel ;
        express:hasString  "0762 x 2032mm" .

inst:IfcPositiveLengthMeasure_17817
        a                  ifcowl:IfcPositiveLengthMeasure ;
        express:hasDouble  "2032."^^xsd:double .

inst:IfcPositiveLengthMeasure_17812
        a                  ifcowl:IfcPositiveLengthMeasure ;
        express:hasDouble  "762."^^xsd:double .

inst:IfcOwnerHistory_41
        a       ifcowl:IfcOwnerHistory ;
        ifcowl:changeAction_IfcOwnerHistory
                ifcowl:NOCHANGE ;
        ifcowl:creationDate_IfcOwnerHistory
                inst:IfcTimeStamp_6416 ;
        ifcowl:owningApplication_IfcOwnerHistory
                inst:IfcApplication_5 ;
        ifcowl:owningUser_IfcOwnerHistory
                inst:IfcPersonAndOrganization_38 .

inst:IfcProductDefinitionShape_10949
        a       ifcowl:IfcProductDefinitionShape ;
        ifcowl:representations_IfcProductRepresentation
                inst:IfcRepresentation_List_17827 .
```

```
inst:IfcIdentifier_17833
        a                   ifcowl:IfcIdentifier ;
        express:hasString   "329774" .

<http://example.com/id/ifc/20190721_122646/GUID/3c0b405a-6a4d-4a42-aeda-
4e012712b48e>
        a                             ifcowl:IfcRelDefinesByType ;
        ifcowl:globalId_IfcRoot       inst:IfcGloballyUniqueId_18126 ;
        ifcowl:ownerHistory_IfcRoot   inst:IfcOwnerHistory_41 ;
        ifcowl:relatedObjects_IfcRelDefines
                <http://example.com/id/ifc/20190721_122646/GUID/145adf16-8b67-460a-
bca8-932e5c98325c> , <http://example.com/id/ifc/20190721_122646/GUID/145adf16-8b67-
460a-bca8-932e5c9832bd> ;
        ifcowl:relatingType_IfcRelDefinesByType
                <http://example.com/id/ifc/20190721_122646/GUID/145adf16-8b67-460a-
bca8-933b5c9832df> .
```

Appendix B. RDF file of Manufacturer data

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix gr: <http://purl.org/goodrelations/v1#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .


<http://example.com/id/doors/D3454> a gr:ProductOrService ;
       gr:name "D3454" ;
       gr:serialNumber "EN658075" ;
       gr:width "762"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "130"^^xsd:double ;
       gr:hasInventoryLevel "15"^^xsd:int .

<http://example.com/id/doors/D3455> a gr:ProductOrService ;
       gr:name "D3455" ;
       gr:serialNumber "EN658071" ;
       gr:width "915"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "145"^^xsd:double ;
       gr:hasInventoryLevel "14"^^xsd:int .

<http://example.com/id/doors/D3456> a gr:ProductOrService ;
       gr:name "D3456" ;
       gr:serialNumber "EN658067" ;
       gr:width "813"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "150"^^xsd:double ;
       gr:hasInventoryLevel "0"^^xsd:int .

<http://example.com/id/doors/D3457> a gr:ProductOrService ;
       gr:name "D3457" ;
       gr:serialNumber "EN658063" ;
       gr:width "864"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "165"^^xsd:double ;
       gr:hasInventoryLevel "2"^^xsd:int .

<http://example.com/id/doors/D3458> a gr:ProductOrService ;
       gr:name "D3458" ;
       gr:serialNumber "EN658059" ;
       gr:width "864"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "175"^^xsd:double ;
       gr:hasInventoryLevel "3"^^xsd:int .

<http://example.com/id/doors/D3459> a gr:ProductOrService ;
       gr:name "D3459" ;
       gr:serialNumber "EN658055" ;
       gr:width "915"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "180"^^xsd:double ;
       gr:hasInventoryLevel "5"^^xsd:int .

<http://example.com/id/doors/D3460> a gr:ProductOrService ;
       gr:name "D3460" ;
       gr:serialNumber "EN658051" ;
       gr:width "864"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "170"^^xsd:double ;
       gr:hasInventoryLevel "8"^^xsd:int .

<http://example.com/id/doors/D3461> a gr:ProductOrService ;
```

```
       gr:name "D3461" ;
       gr:serialNumber "EN658047" ;
       gr:width "864"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "175"^^xsd:double ;
       gr:hasInventoryLevel "18"^^xsd:int .

<http://example.com/id/doors/D3462> a gr:ProductOrService ;
       gr:name "D3462" ;
       gr:serialNumber "EN658085" ;
       gr:width "762"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "160"^^xsd:double ;
       gr:hasInventoryLevel "0"^^xsd:int .

<http://example.com/id/doors/D3463> a gr:ProductOrService ;
       gr:name "D3463" ;
       gr:serialNumber "EN658086" ;
       gr:width "864"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "160"^^xsd:double ;
       gr:hasInventoryLevel "24"^^xsd:int .

<http://example.com/id/doors/D3464> a gr:ProductOrService ;
       gr:name "D3464" ;
       gr:serialNumber "EN658080" ;
       gr:width "915"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "175"^^xsd:double ;
       gr:hasInventoryLevel "32"^^xsd:int .

<http://example.com/id/doors/D3465> a gr:ProductOrService ;
       gr:name "D3465" ;
       gr:serialNumber "EN658076" ;
       gr:width "762"^^xsd:double ;
       gr:height "2032"^^xsd:double ;
       gr:hasPriceSpecification "155"^^xsd:double ;
       gr:hasInventoryLevel "0"^^xsd:int .

<http://example.com/id/doors/D3466> a gr:ProductOrService ;
       gr:name "D3466" ;
       gr:serialNumber "EN658089" ;
       gr:width "762"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "185"^^xsd:double ;
       gr:hasInventoryLevel "10"^^xsd:int .

<http://example.com/id/doors/D3467> a gr:ProductOrService ;
       gr:name "D3467" ;
       gr:serialNumber "EN658092" ;
       gr:width "864"^^xsd:double ;
       gr:height "2134"^^xsd:double ;
       gr:hasPriceSpecification "190"^^xsd:double ;
       gr:hasInventoryLevel "0"^^xsd:int .
```

Appendix C. Proof of concept queries

Scenario 1:

Query 1. Requesting a list of the doors included in the Revit model

```
PREFIX ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX express: <https://w3id.org/express#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT Distinct  ?doorD ?IfcIdentifier ?hD ?wD

WHERE {
  ?doorD a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifcowl:tag_IfcElement ?tag.
  ?tag express:hasString ?IfcIdentifier.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
}
```

Query 2. Requesting available door type options for set of dimensions

```
PREFIX ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX express: <https://w3id.org/express#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT distinct ?hS ?wS (COUNT(*)/2 AS ?DoorsNeeded) ?doortypecode ?DoorsAvailable
?UnitPrice

WHERE {
  ?doorD a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifc:name_IfcRoot ?root.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
  ?panel ifc:name_IfcRoot ?root.
  ?doorStyle ifc:hasPropertySets_IfcTypeObject ?panel.
  ?typedef ifcowl:relatingType_IfcRelDefinesByType ?doorstyle;
           ifcowl:relatedObjects_IfcRelDefines ?doorD;
           ifcowl:relatedObjects_IfcRelDefines ?IfcDoor.
Service <http://10.99.99.4:7200/repositories/exceldata> {
?doorS gr:serialNumber ?doortypecode;
    gr:width ?wS;
    gr:height ?hS;
    gr:hasPriceSpecification ?UnitPrice;
    gr:hasInventoryLevel ?Inventory.
    }
Filter (?wD = ?wS && ?hS = ?hD).
Bind (?Inventory AS ?DoorsAvailable).
}
GROUP BY ?wS ?hS ?DoorsAvailable ?doortypecode ?UnitPrice
```

Scenario 2:

Query 3: Requesting a list of the door types designed, the amount of every type, their availability and their unit price

```
PREFIX ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX express: <https://w3id.org/express#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT distinct ?doortypecode (COUNT(*)/2 AS ?DoorsNeeded) ?DoorsAvailable
?UnitPrice
WHERE {
  ?doorD a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifc:name_IfcRoot ?root.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
  ?panel ifc:name_IfcRoot ?root.
  ?doorStyle ifc:hasPropertySets_IfcTypeObject ?panel;
             ifc:name_IfcRoot [express:hasString ?inputcode].
  ?typedef ifcowl:relatingType_IfcRelDefinesByType ?doorstyle;
           ifcowl:relatedObjects_IfcRelDefines ?doorD;
           ifcowl:relatedObjects_IfcRelDefines ?IfcDoor.
Service <http://10.99.99.4:7200/repositories/exceldata> {
?doorS a gr:ProductOrService;
    gr:name ?name;
    gr:serialNumber ?doortypecode;
    gr:width ?wS;
    gr:height ?hS;
       gr:hasPriceSpecification ?UnitPrice;
       gr:hasInventoryLevel ?Inventory.
    }
filter (?wS = ?wD && ?hS = ?hD && ?doortypecode = ?inputcode).
Bind (?doorD AS ?DoorsNeeded).
Bind (?Inventory AS ?DoorsAvailable).
}
GROUP BY ?doortypecode  ?DoorsAvailable ?UnitPrice
```

Query 4: Requesting available alternative door types

```
PREFIX ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX express: <https://w3id.org/express#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ifcowl: <http://standards.buildingsmart.org/IFC/DEV/IFC2x3/TC1/OWL#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX gr: <http://purl.org/goodrelations/v1#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT Distinct  ?reldoor ?hD ?wD ?CurrentType ?AltType ?amount
WHERE {
  ?door a ifc:IfcDoor;
  ifc:overallHeight_IfcDoor ?h;
  ifc:overallWidth_IfcDoor ?w;
  ifc:name_IfcRoot ?root.
  ?h <https://w3id.org/express#hasDouble> ?hD.
  ?w <https://w3id.org/express#hasDouble> ?wD.
```

```
  ?panel ifc:name_IfcRoot ?root.
  ?doorStyle ifc:hasPropertySets_IfcTypeObject ?panel;
             ifc:name_IfcRoot [express:hasString ?inputcode].
  ?typedef ifcowl:relatingType_IfcRelDefinesByType ?doorstyle;
           ifcowl:relatedObjects_IfcRelDefines ?door;
           ifcowl:relatedObjects_IfcRelDefines ?reldoor.

Service <http://192.168.1.4:7200/repositories/exceldata> {
?doorS a gr:ProductOrService;
    gr:name ?name;
    gr:serialNumber ?doortypecode;
    gr:width ?wS;
    gr:height ?hS;
       gr:hasPriceSpecification ?price;
       gr:hasInventoryLevel ?amount.
    }
filter (?wS = ?wD && ?hS = ?hD && ?inputcode != ?doortypecode && ?amount != 0.0 ).
Bind (?inputcode AS ?CurrentType).
Bind (?doortypecode AS ?AltType).
}
```