Experiments of tools IFC to Ifcowl and IFC to LBD

Fatemah Alhamdoosh
Università degli studi di Firenze
fatemah.alhamdoosh@stud.unifi.it

April 26, 2022

1 Introduction

Building information can be divided logically into many categories such as :
1-the identity and semantics: names of the objects (beam, door, point, ...), machine-readable unique identifier (id code to identify a certain object in the building) , object type or function (electrical, mechanical, civil ...)
3-the characteristics or attributes: material, color, and thermal properties, ... ,ect.
4-objects : products (columns, beam, slabs, door, window,..)
4- relationships between objects: locations (positions express in cartesian points..), connections (contains, above , under and ownership)
5- abstract concepts like performance and costing,
6-processes such as installation and operations(loading beam..)
7- people (owners, designers, contractors, suppliers, etc.).
8- ...
and much more concepts.
Industry Foundation Classes (IFC) is a global standard for data exchange in the building industries. IFC is both a common data model and an open file format. Building industry professionals can use IFC to share data regardless of what software application they use to get their job done. Similarly data from one phase of the building lifecycle can be utilised in a later stage without the need for data re entry, custom import interfaces or proprietary plugins. By providing ifc export and import interfaces that conforms with the IFC standard(s) the application vendor are able to provide interoperability with hundreds of other BIM tools and domain applications.Figure 1 shows how Ifc file looks in BIM software like usBim platform. IFC schema was defined as an EXPRESS schema in which a number

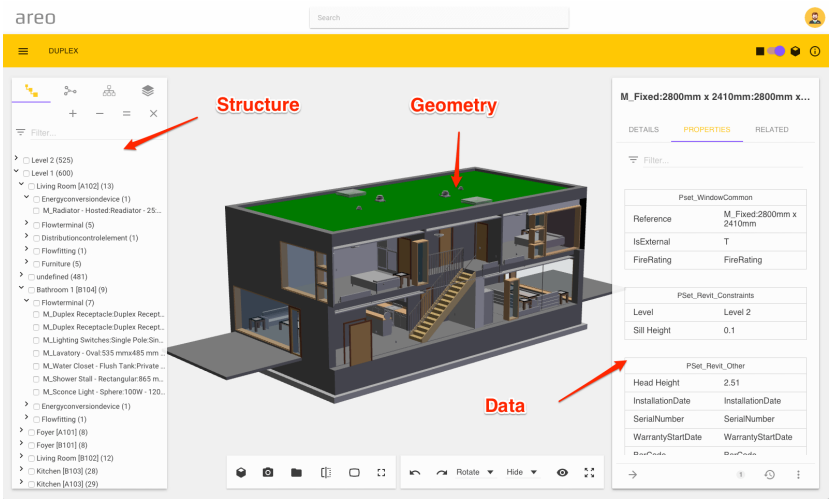

Figure 1: IFC models contain a structures combination of geometric and non-gemetric data. This data can be displayed, analysed and modified in different ways in multiple software applications

of declarations can be made like Type, Entity, Schema, Constant, Rule and Algorithm (Function and Procedural declarations). Figure 2 shows a piece of IFC schema including some declarations of Entity, Type and Function. While an IFC file contains a number of instances of declarations. Each instance has an id which can be used by other instances to create a whole object. An example would be looks like the following:
#25951= IFCCARTESIANPOINT((9513.10035408592,286.015318081989,6200.));
#28574= IFCFACE((#28573));
#132471= IFCBEAM('27AleUoxrCCvsnsoHgCpo$',#42,'UB-Universal Beam:356x171x51UB:264048',$,'UB-Universal Beam:356x171x51UB:71875',#132447,#132467,'264048',.BEAM.);

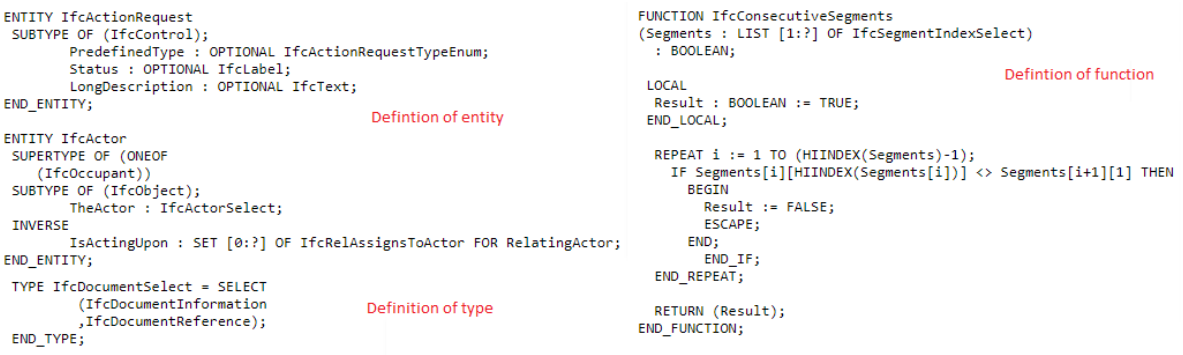

Figure 2: An example of declarations (Entity, type and function) in IFC schema.

2 IFC to Ifcowl

ifcOWL provides a Web Ontology Language (OWL) representation of the Industry Foundation Classes (IFC) schema. Using the ifcOWL ontology, one can represent building data using state of the art web technologies (semantic web and linked data technologies). IFC data thus becomes available in directed labelled graphs (RDF). This graph model and the underlying web technology stack allows building data to be easily linked to material data, GIS data, product manufacturer data, sensor data, classification schema, social data, and so forth. The result is a web of linked building data that brings major opportunities for data management and exchange in the construction industry and beyond. IFCOWL was continually maintained by building smart The transition from IFC to OWL is not unique and each author has adapted the conversion to the specific needs of the user. In the concrete case of lifting the EXPRESS schema of the IFCs onto an ontological level, the mapping between class hierarchy compositions in EXPRESS and the subsumption operator in the DL underlying OWL is rather straightforward: For each ENTITY definition in the schema a corresponding owl:Class is created as a concept in the TBox of the ontology. SUBTYPE OF and SUPERTYPE OF relations are transformed into rdfs:subClassOf relations. There are, however, certain aspects of the composition mechanisms available in EXPRESS that are hardly transformable into OWL:although a SUPERTYPE OF construct prepended by the ABSTRACT keyword prevents the instantiation of a particular class, no direct mechanism in OWL exists that, for example, prevents the assertion of some resource to be a mere "IfcObject". For the simple data types in EXPRESS such as Integer, Real, String, and so forth,it was suggests the creation of wrapper classes and then subclass defined types from these wrappers. Each of the wrapper classes has a single owl:DatatypeProperty with a range of the according XML Schema type.
To limit the range of properties to some concrete individuals or simple data type values, OWL offers the owl:oneOf construction. For the many cases where the nature of some specific concept is further specified by an enumeration or concrete value in the IFC model or other STEP-based models, like setting the general construction type of a roof to one of "FLAT_ROOF", "SHED_ROOF", "GABLE_ROOF", and so forth, via the "IfcRoofTypeEnum", the range of the corresponding attribute is set to one of the string values that represents it.ifcOwl presents a lot of typeEnum, figure 3 shows an example.

Among all the important classes, the IfcRoot class is the most important and has a deep hierarchy.it

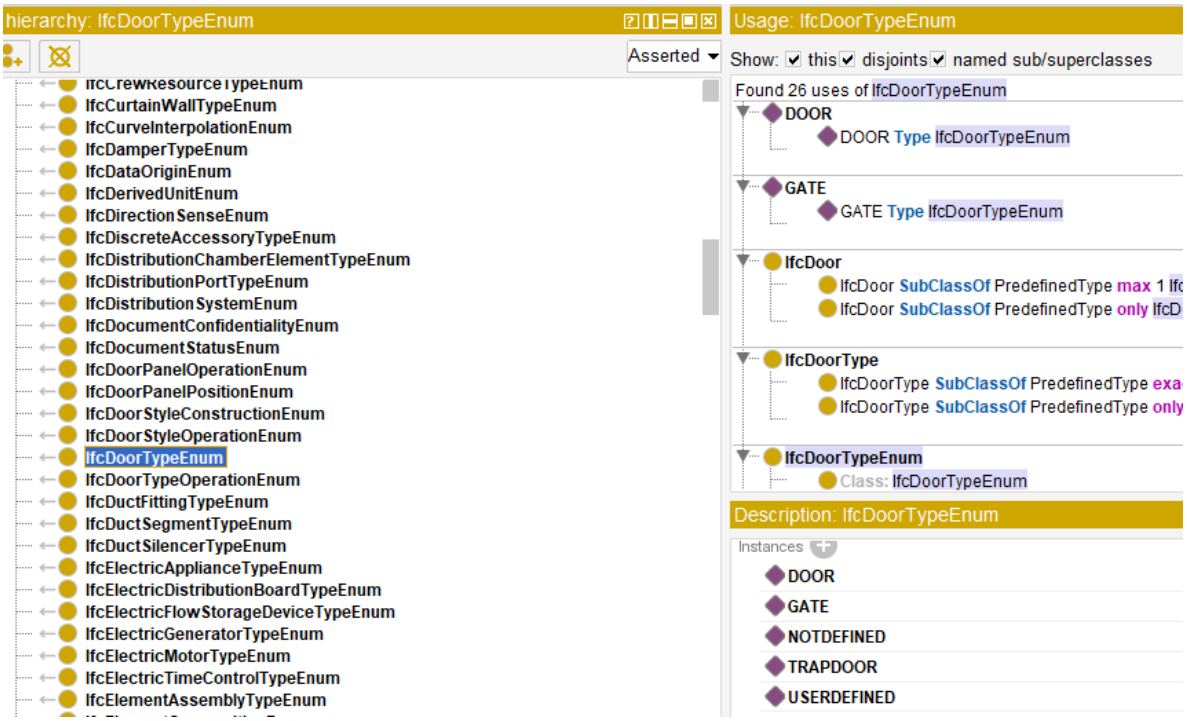

Figure 3: A screenshot of protegè viewing a number of subclasses of the Enumeration class, showing at the bottom left the possible door types

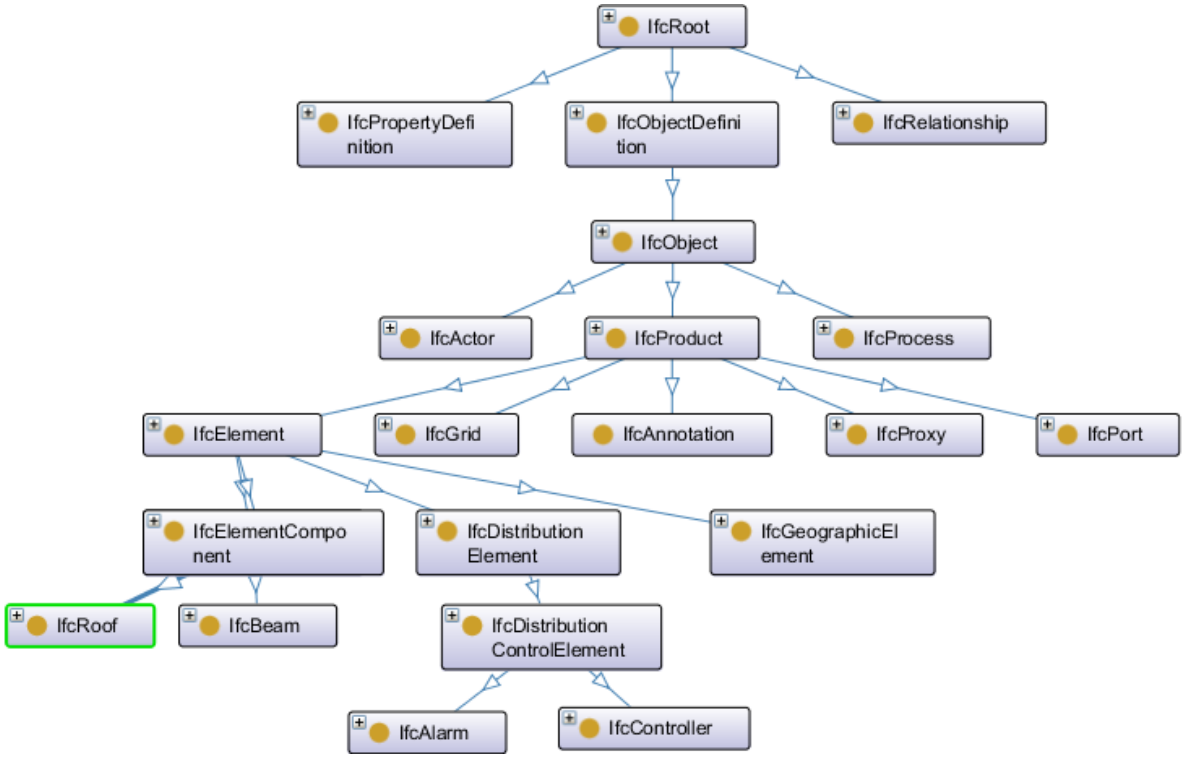

Figure 4: This is some of subclasse in the ifcroot showing using OntoGragh tool in protegè. The ifcProduct has the deeper hierarchy.

can be instantiant just one time in the file. Figure 4 shows some of the subclasses of IfcRoot. It has three principal subclasses: IfcObjectDefinition, IfcPropertyDefinition and IfcRelationship. IfcObject-

Definition is a class that explain all information about an object. it has three subclasses: IfcContext, IfcObject and IfcTypeObject. IfcObject has the following subclasses: IfcActor, IfcControl, IfcGroup, IfcProcess, IfcProduct and IfcResource. By this classes we represents a various types of objects, from single products to group of objects and all the relative information of control, processing and resource. Figure 5 shows an example of some triples in turtle format obtained using the converter IFC to IfcOwl, inst:ifcBeam-58763 is beam of type IfcBeam with many proprety. we focus on the property object-Placement_ifcProduct which hold a formation about the beam location, showing how ifcowl describe in detailed and abstracted way the building information.

```
inst:IfcBeam_58763 rdf:type owl:NamedIndividual , (1)
                ifc:IfcBeam ;
          ifc:globalId_IfcRoot inst:IfcGloballyUniqueId_117963 ;
          ifc:name_IfcRoot inst:IfcLabel_117964 ;
          ifc:objectPlacement_IfcProduct inst:IfcLocalPlacement_58739 ;
          ifc:objectType_IfcObject inst:IfcLabel_101207 ;
          ifc:ownerHistory_IfcRoot inst:IfcOwnerHistory_42 ;
          ifc:predefinedType_IfcBeam ifc:BEAM ;
          ifc:representation_IfcProduct inst:IfcProductDefinitionShape_58759 ;
          ifc:tag_IfcElement inst:IfcIdentifier_117965 .
inst:IfcLocalPlacement_58739 rdf:type owl:NamedIndividual , (2)
                ifc:IfcLocalPlacement ;
          ifc:placementRelTo_IfcLocalPlacement inst:IfcLocalPlacement_148 ;
          ifc:relativePlacement_IfcLocalPlacement inst:IfcAxis2Placement3D_58738 .
inst:IfcAxis2Placement3D_58738 rdf:type owl:NamedIndividual , (3)
                ifc:IfcAxis2Placement3D ;
          ifc:axis_IfcAxis2Placement3D inst:IfcDirection_20 ;
          ifc:location_IfcPlacement inst:IfcCartesianPoint_58734 ;
          ifc:refDirection_IfcAxis2Placement3D inst:IfcDirection_58736 .
inst:IfcDirection_20 rdf:type owl:NamedIndividual , (4)
                ifc:IfcDirection ;
          ifc:directionRatios_IfcDirection inst:IfcReal_List_100654 .
inst:IfcReal_List_100654 rdf:type owl:NamedIndividual , (5)
                ifc:IfcReal_List ;
          list:hasContents inst:IfcReal_100643 ;
          list:hasNext inst:IfcReal_List_100655 .
inst:IfcReal_100643 rdf:type owl:NamedIndividual , (6)
                ifc:IfcReal ;
          express:hasDouble "0.0"^^xsd:double .
```

Figure 5: An example of some triples in turtle format obtained using the converter IFC to IfcOwl. showing that we need a 6 level to represents a direction of a beam.

3 IFC to LBD

IfcOWL overcomes some of IFC drawbacks such as ifcOWL makes the extension of the schema much more easily as web ontologies are modular. IfcOWL permits linking of data between vary disciplinary while ifc limits the linking on files containers. IFC dose not provide standard querying and reasoning, while ifcOWL do.
However, IfcOWL has over 1300 classes and over 1500 object property and has extremely complex structure which yield in complex ABox which contains a lot of redundant triples like geometry information affecting negatively on the query writing and consequently on the performance of inference.
As it was stated in [lbd], a more generic approach to convert IFC files to OWl using modular ontologies is seen as a necessity for real industrial applications.
Ontology modularization is the task of partitioning an ontology splitting up the set of axioms into a set of modules $(M_1, ...M_k)$ such the each M is an ontology and the union of all modules is semantically equivalent to the original ontology O. Modular ontology has a lot of benefits as it reduces the complexity over vary phases: design phase, deployment, management and usage. It also allows to reuse the same ontology in several different contexts.
IFC To LBD converter basically convert IFC to modular Linked Building Data (LBD) graphs, aiming initially at the BOT, PROPS and PRODUCT ontologies. Relevant information from IFC building models is extracted and transformed Using such Abox modules, which can be used independently from each other, it becomes easier to understand the structure of the combined graph.
The Building Typology Ontology (BOT) was developed within the W3C LBD CG as a central and modular ontology for the AEC industry, followed by the emergence of subgroups developing a whole range of other modular ontologies for building products, building-related properties, geometry, etc.
BOT is developed to allow defining a building's topology, by using dedicated BOT classes (bot:Site, bot:Building, bot:Storey, bot:Space and bot:Element) and specific BOT relations between them (e.g. bot:containsElement).
The PRODUCT ontology is designed to classify individual building objects, e.g. as a p4bldg:Wall. Figur 6 shows the classes hierarchy in bot ontology and it's relations. Shows also some classes in Product ontology.
The PROPS ontology is used to assign properties to building-related elements. As the PROPS ontology is still in conceptual design phase, no ontology (Tbox) is available yet. The current early proposal for the PROPS ontology structure includes three levels of complexity (L1,L2 and L3).
Figure 7 shows an example of a new construction of a slab and the different complexity level in three ontology L1, L2 and L3. Props L3 is the most detailed one which use proprieties from another ontologies (seas and schema).

If the user selects the PROPS module, he can choose one of the three PROPS levels of complexity. If PROPS L2 or L3 is selected, the user can choose to define the in between property instance nodes (L2 and L3) and/or the state instance nodes (L3) as blank nodes or instances with a stable URI. If the above nodes do not have a stable URI, they are not unique outside the database or file they reside in, making it impossible to connect them to other resources outside their original environment. Using blank nodes can decrease the file size of the exported RDF and increase the human readability of the Turtle file. By using the corresponding classes between the ifcOWL and the LBD modular ontologies (BOT, PRODUCT, PROPS) displayed in Table 1 in the [Pau], the newly created instance resources get the right LBD OWL classes assigned.
The use of Abox modules make it easier to share and study specific parts of the converted LBD graph. Regarding the PROPS module, users can select the most appropriate level of complexity they need for their project.
[Pau]. Figure 8 shows an example of some triples in turtle format obtained using the converter IFC to LBD. Showing how we can represent a beam using the bot, product, and Props L3. Compared to the similar beam direction representation in IfcOWL, we need just three levels of depth instead of the six required levels in IfcOwl, demonstrating that LBD is much less complex.

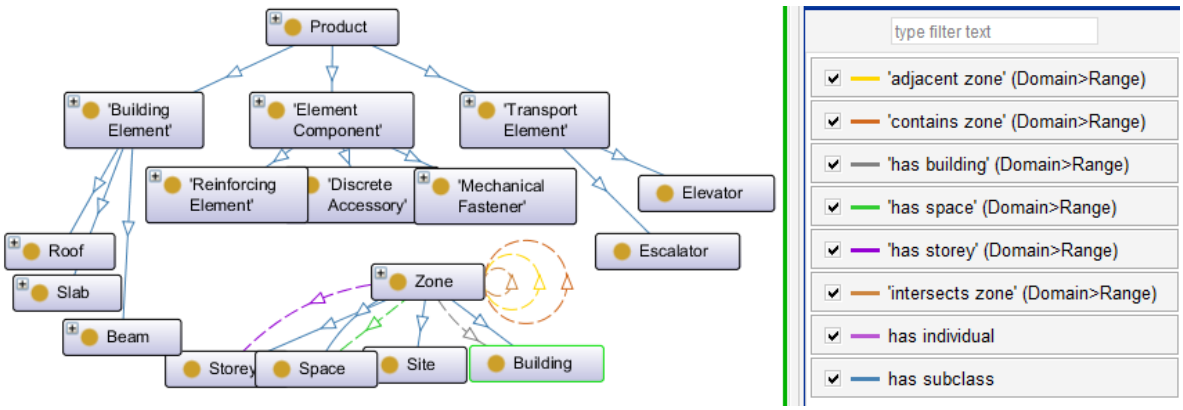

Figure 6: Using Onto-graph tool we show the classes hierarchy in bot ontology and it's relations, showing also some classes in Product ontology.

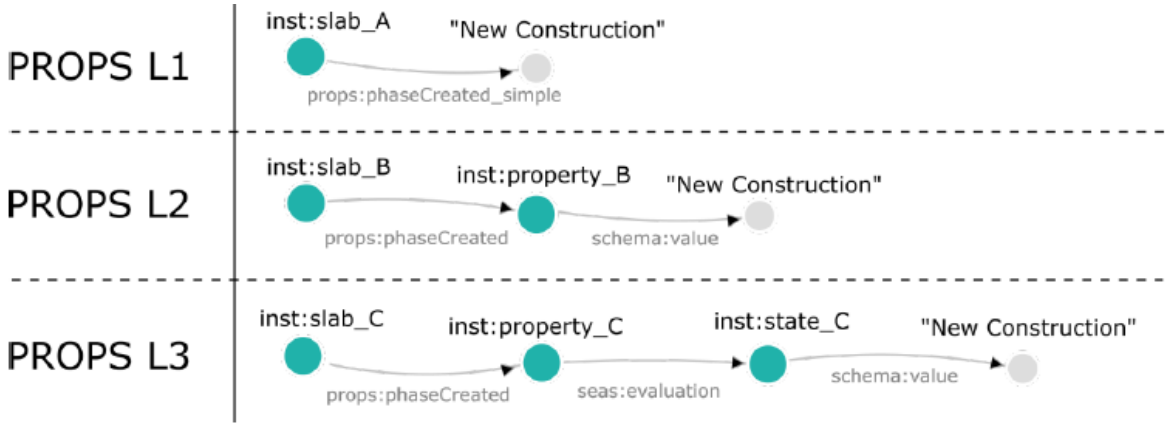

Figure 7: An example of a new construction of a slab and the different complexity level in three ontology Props L1, Props L2 and Props L3. Props L3 is the most detailed one which use proprieties from another ontologies (seas and schema).

4 Applications using linked building data

In this section i will try to explain some usages and benefits of the using of Linked building data. There ara many important application where the use of linked data facilitates task and make the things more comprehensive and simple.
In [EG] authors introdues an approach based on the ifcOWL ontology to support indoor navigation.Its aim is to try to take advan- tage of the increasing use of BIM files for application to indoor navigation, since this kind of file contains useful information in that context. This information should be complemented by data from sensors included in the navigation system. To this end, a series of modifications of this ontology have been proposed that include new items, SWRL ontological deduction rules and SQWRL searches. These extensions can be used to calculate the absolute position of the different elements of the ontology with respect to a global coordinate system, to deduce the possible location of different room types and to process orders for setting navigation destination. By using this ontology, it is possible to interact quickly and easily with other elements within the same knowledge domain.
In [AA] authors developed a method to handle physical sensors through a BIM representation is developed. The proposed method provides a convenient way to select and access monitored data for Building Energy Simulation model validation and constitutes a way to centralize building performance assessment studies around BIM.
Another important application int the field of management buildings has been introduced in [HW]. Authors have presented a system of comprehensive intelligent energy analysis in building. In the developed system, they combined classical data-driven energy analysis with novel knowledge-driven

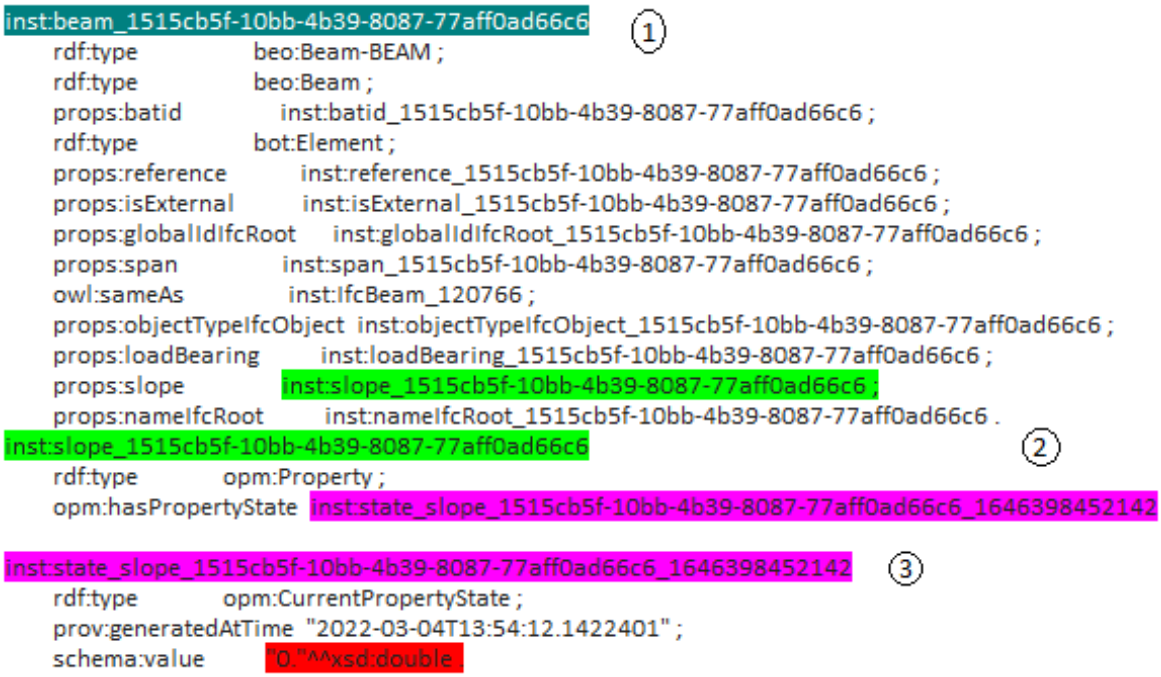

Figure 8: shows an example of some triples in turtle format obtained using the converter IFC to LBD. Showing how we can represent a beam using the bot, product, and Props L3.

energy analysis that supported by ontology. The analysis is performed on information collected from building automation devices. The ontology supported analysis approach provides intelligent assistance to improve energy efficiency in households or public buildings, by strongly considering individual user behavior and current states in the building. Users do not have to read the whole energy consumption data or energy usage profile curves in order to understand their energy usage pattern. The system will understand the energy usage pattern, and notify user when energy inefficient conditions occur.

5 Experiments

The first tool we test is an IFC-to-RDF converter developed by [Pau], The application converts an IFC file to an RDF Abox graph-structured according to ifcOWL, i.e. sets of RDF triples that contain assertions between individual RDF resources. The resulting file is in turtle format (.ttl).
For using the tool, it is necessary to install java and download the shaded file from the page of realises [Pau]. All instructions and essential information are on the page of the tool. It's pretty simple to use. It can be used by integrating it into a java program or by the command line in CLI. An example of a command running in CLI looks like this:

```
java -jar IFCtoRDF-0.4-SNAPSHOT-shaded.jar --dir path/to/folder/
```

The second tool is IFC To LBD developed by [lbd]. They provide a desktop version jar file. However, it failed to run on Linux. However, on Windows with a recent java version worked perfectly. This tool can also convert IFC to ifcOWL file, all files in turtle format.

For testing the two above tools, I searched for IFC files on [mod]. I have chosen many files and converted them to ifcOWL and LBD. However, I could not upload all files into the virtuoso RDF store because of the large dimension of the files. I tried to upload testing files created by the developers of the tools, but I had the same problem and same error when the size of the file exceeded 50MB.

```
You have attempted to upload invalid data. You can only upload RDF, Turtle, N3
```

So I spent a lot of time fixing that problem, but there is no clear, direct and quick tutorial to fix this problem. I tried a bulk upload of files, but I had no success. In the end, I decided to do experiments over the simple files which I cloud uploaded. So on, in this report, I will report and discuss two cases(files), also if, in reality, I tested more files.

5.1 hospital structural and architecture files

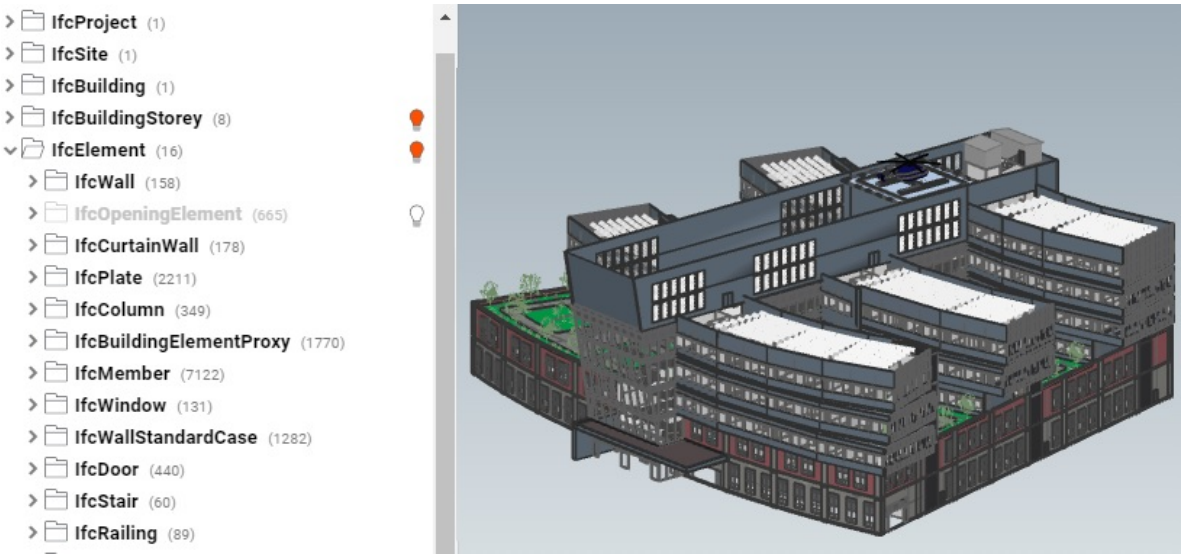

Figure 9: This figure is a visualisation of the hospital architecture IFC file. I showed some of the IFC classes and the corresponding number of instances on the left.

The first case I tested was an IFC file of a hospital structure. The size of the file is 6.3 MB. Table 1 shows some statistical measures. We can note the massive decrease in the size of the RDF graph when using LBD and choosing The PROPS L3. we can obtain more reduction using PROPS L1 and PROPS L2. Figure 12 shows the 24 used classes in the graph using the PROPS L2. The class CurrentPropertyState was omitted when using the PROPS L2. Using the PROPS L1, it deletes the object property " Property". Figure 11 shows the 32 properties used in the resulting graph using LBD with PROPS L3. Using the L2 PROPS, The proprieties generatedAtTime and hasPropertyState are not used. In addition to the two precedent properties, the proprieties "value" ,unit,hasGeometry and asWKT was not used in the file generated using PROPS L1. The omitted properties demonstrate the huge decrease in the size of the file. It's difficult to report the classes and the properties used in the file ifcOWL. In the section SPARQL, it was reported precisely all SPARQL queries used to produce the results.
However, it's easy to note that all properties and classes used in LBD are a subset of equivalent classes used in ifcOWL.
At this point, we do more domain-specific queries to understand the ability and the efficiency of the created RDFs.

For the architecture file, his size is 80Mb, the size of the RDF file using ifcOWL is 708MB, while the size of the RDF file using LBD with the L3 props is just 20MB!!. so I cloud not upload the cowl to virtuoso to make some confront. But on the LBD file, I made the same queries that I used on the structure file.

5.1.1 bot:ContainsElemnts

To know how much and which type of elements of a particular class we have in our RDF that are contained in a class, we perform the following query on the LBD RDF created. It's appropriate to say

	Ifcowl	LBD with L1	LBD with L2	LBD with L3
size of the file	50 MB	2.1MB	6.8 MB	12.4 MB
Number of triples	694688	33522	77925	144501
Number of classes	124	21	23	24
Number of properties	219	26	30	32
For the file of				

Table 1: A tabel that show the size of the files, the number of triples in each file and the number of classes and property used.

that is no direct mapping of the property ContainsElements in the LBD. Rather than there are used multiple properties for each class, and to obtain the same information of the ifcOWL, we need a high knowledge about classes and properties existing in the ontology, which is very complex. The result of the query is in figure 10 .

```
select distinct ?element ?contained_element count(?t1)
where { ?s bot:containsElement ?t1.
?s a ?element.
?t1 a ?contained_element. }
```

```
select distinct ?class ?p ?ob count(?t1)
where { ?s ?p ?t1.
?s a ?class.
?t1 a ?ob.}
```

element	contained_element	count
https://w3id.org/bot#Storey	https://w3id.org/bot#Element	2825
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Beam	1970
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Beam-BEAM	1970
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Footing-NOTDEFINED	535
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Footing	553
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Column	255
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Column-COLUMN	255
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Wall-NOTDEFINED	28
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Wall	28
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Slab	11
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Slab-FLOOR	11
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Footing-STRIP_FOOTING	15
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Member	5
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Member-BRACE	5
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Railing-NOTDEFINED	2
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Railing	2
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Stair	1
https://w3id.org/bot#Storey	https://pi.pauwel.be/voc/buildingelement#Stair-NOTDEFINED	1

Figure 10: A Table that shows the domain and the range of the property ContainsElemnts and the number of triples for each couple domain range.

5.1.2 Retrieve information about the height and the width of doors

Another important query that one could make is to query about the height and the width existing combinations of the doors.It maybe useful in the production level in order to answer the question: how

many doors of a certain dimension we need to order?
Figur 13 show results . a many other similar informations can be obtained by replacing a door with window , wall, railing ..ect.

```
 PREFIX buildingelement:<https://pi.pauwel.be/voc/buildingelement#>
PREFIX props:<http://lbd.arch.rwth-aachen.de/props#>
PREFIX owl:<https://w3id.org/def/smls-owl#>

select ?valueWidth ?unitwidth ?valuehieght ?unithieght (count(?s) as ?count)
where { ?s a buildingelement:Door.
?s props:overallHeightIfcDoor ?o11;
props:overallWidthIfcDoor ?o22.
?o22 ?p22 ?o2.
?o11 ?p11 ?o1.
?o2 schema:value ?valueWidth;owl:unit ?unitwidth.
?o1 schema:value ?valuehieght;owl:unit ?unithieght.

 }group by ?valueWidth ?unitwidth ?valuehieght ?unithieght order by desc (?count)
```

5.2 Other Sparql query

In this section i report some sparql query that i have used to understand an validate the data, without describing the results.

5.2.1 Get the number of tripls

```
 select count(*) where { ?s ?p ?o}
```

5.2.2 count number of classes and properties

Classes

```
select (count(distinct ?o) as ?count) where { ?s rdf:type ?o}
```

Properties

```
select (count(distinct ?p) as ?count) where { ?s ?p ?o}
```

5.2.3 Count number of triple for each proprety and classes

Classes

```
select ?o (count(distinct ?s) as ?count) where { ?s rdf:type ?o} group by(?o)order by desc
    (?count )
```

Properties

```
select ?p (count(*) as ?count) where { ?s ?p ?o} group by(?p)order by desc (?count )
```

5.2.4 others

```
select distinct ?q ?w where { ?s ?p ifc:IfcCircle.
?s ?c ?d.
?d a ifc:IfcPositiveLengthMeasure.
?d ?q ?w}
```

To get all the proprety and entity paths of beam object using ifcowl.

```
select distinct * where {
<https://www.ugent.be/myAwesomeFirstBIMProject#IfcBeam_891> ?p ?o.
?o ?p1 ?o1.
?o1 ?p2 ?o2.
?o2 ?p3 ?o3.
?o3 ?p4 ?o4.
?o4 ?p5 ?o5.
?o5 ?p6 ?o6}
```

To get all the same above using LBD.

```
select distinct * where { ?s a <https://pi.pauwel.be/voc/buildingelement#Beam>.
?s ?p ?o.
?o ?p1 ?o1.
?o1 ?p2 ?o2.
}
```

```
select * where{ ?s a buildingelement:Beam.
?s props:isExternal ?o.
?o opm:hasPropertyState ?o2.
?o2 schema:value ?value.
Filter(?value =0)
}
```

```
select ?value (count(?value) as ?count) where{ ?s a buildingelement:Beam.
?s props:span ?o.
?o opm:hasPropertyState ?o2.
?o2 schema:value ?value.
}group by ?value order by desc (?count)
```

In this query we try to obtain all legnth of a the exsistings beams in the models using LBD.

```
Prefix buildingelement:<https://pi.pauwel.be/voc/buildingelement#>
PREFIX props:<http://lbd.arch.rwth-aachen.de/props#>
Prefix opm:<https://w3id.org/opm#>

select ?value (count(?value) as ?count) where{ ?s a buildingelement:Beam; props:span ?o.
?o opm:hasPropertyState ?o2.
?o2 schema:value ?value.
filter(?value >6000)
}group by ?value order by desc (?count)
```

In the following query we try to obtain the same information above but using ifcowl modelling, The legnth of the path demonstrate that querying in LBD is much simpler and more direct than querying in Ifcowl.

```
  prefix ifcowl:<https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD2/OWL#>
prefix list:<https://w3id.org/list#>

select ?value (count(?value) as ?count) where { ?s a ifcowl:IfcBeam;
    ifcowl:objectPlacement_IfcProduct ?o.
?o ifcowl:relativePlacement_IfcLocalPlacement ?o1.
?o1 ifcowl:location_IfcPlacement ?o2.
?o2 ifcowl:coordinates_IfcCartesianPoint ?o3.
?o3 list:hasNext ?o4.
?o4 list:hasContents ?o5.
?o5 express:hasDouble ?value} group by ?value order by desc (?count)
```

```
 prefix ifcowl:<https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD2/OWL#>
prefix list:<https://w3id.org/list#>

select ?value (count(?value) as ?count) where { ?s a ifcowl:IfcBeam;
    ifcowl:representation_IfcProduct ?o.
?o ifcowl:representations_IfcProductRepresentation ?o1.
?o1 list:hasNext ?o2.
?o2 list:hasContents ?o3.
?o3 ifcowl:items_IfcRepresentation ?o4.
?o4 ifcowl:depth_IfcExtrudedAreaSolid ?o5.
?o5 express:hasDouble ?value} group by ?value order by desc (?value)
```

6 Conclusion

As a resuming of this study i can conclude that LBD model is more efficient, easy and simple.It is also a memory effective model. As seen in this report, i studied the IFC model and two existing tools to convert it into RDF graph. I got in touch with many technologies like ontology, the semantic web tools like sparql language, the usage of Rdf store like virtuoso, and studied a many domain specific application of linked building data. In future working one cloud study the case of large rdf file, and more detailed applications and usage.

References

[AA] Dirk Saelens Ando Andriamamonjy, Ralf Klein. Sensor handling in building information models. development of a method and application on a case study.

[EG] Jonay Toledo Rafael Arnay Leopoldo Acosta Evelio González, José Demetrio Piñeiro. An approach based on the ifcowl ontology to support indoor navigation.

[HW] Polina Häfner Sven Rogalski Hendro Wicaksono, Preslava Dobreva. Methodology to develop ontological building information model for energy management system in building operational phase.

[lbd] Ifctolbd converter. https://github.com/pipauwel/IFCtoRDF.

[mod] http://openifcmodel.cs.auckland.ac.nz/Model.

[Pau] Pieter Pauwels. Ifctoowl converter. https://github.com/pipauwel/IFCtoRDF.

p	count
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	52874
http://schema.org/value	22192
http://www.w3.org/ns/prov#generatedAtTime	22192
https://w3id.org/opm#hasPropertyState	22192
http://lbd.arch.rwth-aachen.de/props#globalIdIfcRoot	2842
http://lbd.arch.rwth-aachen.de/props#nameIfcRoot	2841
https://w3id.org/bot#containsElement	2825
http://lbd.arch.rwth-aachen.de/props#batid	2825
http://lbd.arch.rwth-aachen.de/props#objectTypeIfcObject	2825
http://lbd.arch.rwth-aachen.de/props#reference	2273
http://lbd.arch.rwth-aachen.de/props#isExternal	2272
http://lbd.arch.rwth-aachen.de/props#loadBearing	2269
http://lbd.arch.rwth-aachen.de/props#slope	1984
http://lbd.arch.rwth-aachen.de/props#span	1975
http://lbd.arch.rwth-aachen.de/props#extendToStructure	28
https://w3id.org/def/smls-owl#unit	16
https://w3id.org/bot#hasStorey	15
http://lbd.arch.rwth-aachen.de/props#aboveGround	15
http://lbd.arch.rwth-aachen.de/props#elevationIfcBuildingStorey	15
http://lbd.arch.rwth-aachen.de/props#longNameIfcSpatialElement	15
http://lbd.arch.rwth-aachen.de/props#pitchAngle	4
http://lbd.arch.rwth-aachen.de/props#height	2
https://w3id.org/bot#hasBuilding	1
http://lbd.arch.rwth-aachen.de/props#refElevationIfcSite	1
http://www.opengis.net/ont/geosparql#hasGeometry	1
http://lbd.arch.rwth-aachen.de/props#nosingLength	1
http://lbd.arch.rwth-aachen.de/props#numberOfRiser	1
http://lbd.arch.rwth-aachen.de/props#numberOfTreads	1
http://lbd.arch.rwth-aachen.de/props#riserHeight	1
http://lbd.arch.rwth-aachen.de/props#treadLength	1
http://lbd.arch.rwth-aachen.de/props#numberOfStoreys	1
http://www.opengis.net/ont/geosparql#asWKT	1

Figure 11: LBD object properties (24 in total) with their frequency in the file LBD using PROPSL3

o	count
https://w3id.org/opm#CurrentPropertyState	22192
https://w3id.org/opm#Property	22192
https://w3id.org/bot#Element	2825
https://pi.pauwel.be/voc/buildingelement#Beam	1970
https://pi.pauwel.be/voc/buildingelement#Beam-BEAM	1970
https://pi.pauwel.be/voc/buildingelement#Footing	553
https://pi.pauwel.be/voc/buildingelement#Footing-NOTDEFINED	535
https://pi.pauwel.be/voc/buildingelement#Column	255
https://pi.pauwel.be/voc/buildingelement#Column-COLUMN	255
https://pi.pauwel.be/voc/buildingelement#Wall-NOTDEFINED	28
https://pi.pauwel.be/voc/buildingelement#Wall	28
https://w3id.org/bot#Storey	15
https://pi.pauwel.be/voc/buildingelement#Footing-STRIP_FOOTING	15
https://pi.pauwel.be/voc/buildingelement#Slab	11
https://pi.pauwel.be/voc/buildingelement#Slab-FLOOR	11
https://pi.pauwel.be/voc/buildingelement#Member	5
https://pi.pauwel.be/voc/buildingelement#Member-BRACE	5
https://pi.pauwel.be/voc/buildingelement#Railing-NOTDEFINED	2
https://pi.pauwel.be/voc/buildingelement#Railing	2
https://w3id.org/bot#Building	1
https://pi.pauwel.be/voc/buildingelement#Stair	1
https://pi.pauwel.be/voc/buildingelement#Stair-NOTDEFINED	1
https://w3id.org/bot#Site	1
http://www.opengis.net/ont/geosparql#Feature	1

Figure 12: LBD Properties with their frequency using PROPS L2.

valueWidth	unitwidth	valuehieght	unithieght	count
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	82
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	60
1830.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	44
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	38
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	27
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	26
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	18
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	18
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	17
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	12
1830.0	http://qudt.org/vocab/unit/MilliM	1981.0	http://qudt.org/vocab/unit/MilliM	12
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	9
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	7
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	6
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	5
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	5
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	5
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	4
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	4
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	4
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	4
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	3
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	3
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	2
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	2
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	2
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	2
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	2
2400.0	http://qudt.org/vocab/unit/MilliM	3100.0	http://qudt.org/vocab/unit/MilliM	2
1730.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	1
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	1
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	1
813.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	1
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	1
915.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	1
915.0	http://qudt.org/vocab/unit/MilliM	2032.0	http://qudt.org/vocab/unit/MilliM	1
1730.0	http://qudt.org/vocab/unit/MilliM	2134.0	http://qudt.org/vocab/unit/MilliM	1

Figure 13: Number of existing doors of a certain width and height with unit measure.