Technical University of Munich

Department of Civil, Geo and Environmental Engineering

Chair of Computational Modeling and Simulation

IFC Extension: Multiple Levels of Development Model Container

IDP

for the Master of Science Informatics

Author: Peeranut Chindanonda

Supervisor: Prof. Dr.-Ing. André Borrmann

Advisor: M.Sc. Jimmy Abualdenien

Date: 29. April 2019

Table of Contents

List of Figures III

List of Tables IV

List of Code V

List of Abbreviations VI

1 Introduction and Motivation 7

2 Background and Related Works 9

3 Solutions 12

3.1 To have single IFC file containing multiple LOD. 12

3.2 Each IFC file contain relationships or links to other IFC file. 12

3.4 To put all IFC files from different LOD and another file containing relationships into a single zip file.....15

3.5 Comparing Solutions.....18

4 Software Prototype 19

4.1 Libraries Used.....19

4.2 Usage 20

5 Conclusion and Outlook 26

Bibliography 27

List of Figures

Image 1 User Interface for selecting LOD file 20
Image 2 User interface for viewing refinements 20
Image 3 User interface for visualizing refinements 21
Image 4 Export Linked Multi LOD IFC Zip 22
Image 5 Content of Linked Multi LOD IFC Zip 22
Image 6 Export Multi LOD IFC Zip 24
Image 7 Content of Multi LOD IFC Zip 24

List of Tables

Table 1 Comparison of Solutions..... 18

List of Code

Code 1 Example of IfcExternalObject 13
Code 2 Example of IFC4 XML with refines relationship 13
Code 3 IfcRelRefines Express specification 14
Code 4 Example of IfcExternalObject with relative path 14
Code 5 Example of metadata_v1.json for IFC-SPF 15
Code 6 Example of metadata_v1.json for IFC-XML 16
Code 7 Example of refinements_v1.json 16
Code 8 Example IfcWall with refines relationship 23
Code 9 Example of metadata_v1.json 25
Code 10 Example of refinements_v1.json 25

List of Abbreviations

LOD	Levels of Development
IFC	Industry Foundation Classes
BIM	Building Information Modelling
RDF	Resource Description Framework
URI	Uniform Resource Identifier
XML	Extensible Markup Language
JSON	JavaScript Object Notation
GML	Geography Markup Language
RDF	Resource Description Framework

1 Introduction and Motivation

Designing buildings are the process which involve many collaborative activities. Building Information Modeling (BIM) will have an important role to help the design process systematically. BIM is a digital representation of all information of the building, and when information become digital, it will be easier to communicate, transfer and collaborate. There is a problem that data in different applications are in different formats which make it's hard to exchange data. For this reason, Industry Foundation Classes (IFC) was published as ISO standard in 2003 which is an open neutral standard data format to describe BIM. IFC provides a set of definitions for all object elements such as all stages of a building projects life cycle including a large set of building information representations, geometry design and semantic object model. IFC are used in the building industry and use the concept of objects to describe the model. IFC have multiple format to support different encoding such as IFC-SPF, IFC-XML and IFC-ZIP [1].

However, BIM model are complex and large, to handle the model, it will need people at different specialization. A Model will be changed several times in order to satisfy both various design and engineering requirements. In order to satisfy requirement in reliable way, Levels of Development (LOD) is the part of the model. LOD have generally 5 stages begin with conceptual design until built. LOD will describe model such as usability and limitations of the modelled elements and help specify the needs at every stage.

Nowadays, managing multiple levels of development become an issue to discuss in model-based planning techniques because of the limitation when integrating building model. In order to collaborate and exchange data in different LOD, it should have reliable way or standard to describe and connect data together. According to [2], the exchange data between multiple LOD is a key for collaboration process.

Linking multiple LOD together works by linking every component in the LOD to another component at the next LOD. When components are linked together, it makes analyzing individual component types, visualizing and exchanging easier. The focus of this IDP is to link objects in a level of development to more detailed building element in the next level of development that are in IFC-SPF or IFC-XML format.

2 Background and Related Works

In general, exchanging product model information is an important thing to concern when we talk about interoperability, and to solve this problem, we need a standard that works for all the models. Industry Foundation Classes (IFC) specification which are maintained by buildingSMART describes object-based data model to represent building and civil infrastructure. IFC have been a good solution to solve this problem and there are many software tools that were able to import and export IFC-based file. IFC data can be encoded in various data formats, each have their own advantages and disadvantages. The data formats that are officially released by buildingSMART and actively supported include IFC-SPF and IFC-XML. IFC-SPF are stored in STEP format. It is the most widely used format for IFC and can also be read as plain text. IFC-XML are stored is XML format. It is also stored in plain text and is easier to read by human than IFC-SPF. XML is also more popular format and recognized globally so there are many more software libraries and tools for XML. The disadvantage of XML is that it is usually bigger in size for the same data representation in IFC-SPF format [1].

Another interesting data model is Resource Description Framework (RDF) which already have an IFC ontology described by ifcOWL. RDF is a standard model for data interchange on the Web. RDF can easily be visualized as graph. It also has a query language called SPARQL which are useful for querying objects data.

Another data format that we should consider is JSON. JSON is one of the most popular data formats. It supported structure such as arrays and maps. It is more compact than XML and are supported by libraries from almost every programming languages while also be simple to read.

PM4D Final Report [2] illustrate Product Model and Fourth Dimension approach to construct models using knowledge of building components, geometry and materials. They are using IFC for information sharing and try to use IFC as much as possible for exchanging data to improve the efficiency of information sharing, minimize re-entry of data and to reduce design time during the schematic design phase. However, IFC do not work well in later project phase because of misrepresentation of geometric across different software packages reading the same IFC source file, unstable and unreliable performances, loss of object information in process, and confusion in file revisions. This confusion happens when revising model and exporting updated IFC file after regenerating it which confuse the software that imports it. Therefore, they need an improvement in information exchange standard and more interoperable software in order to get more benefit.

Another standard to describe 3D object is CityGML which is an XML-based encoding for the representation, storage, and exchange of virtual 3D city and landscape models. In [3], they implemented an algorithm to detect matching geometries features within and across LODs and links them but they focus only on 2 geometries which are polygon and linear ring. The first step is to index all the points of linear rings or polygons and then they do the matching to detect geometries that are equivalent or partially equivalent and then consolidate by analysing matched geometry and removing redundant geometries by linking it to other existing geometry. To link and store relationship of topology they use XML Linking Language (XLink) as it is the standard way of linking nodes for XML. The main goal of the approach proposed by the authors of [3] is to reduce file size of CityGML.

Multimodels - Instant nD-Modeling using original Data [4] describes a way to store link data between heterogeneous models by referencing IDs. This allows existing data files to be stored as is, without rewriting into new data schema. This allows multiple applications to store cross-domain data simultaneously easily. This is similar to the approach in solution 3 in this document.

Refinement checking software library are used from *A meta-model approach for formal specification and consistent management of multi-LOD building models* [5] to create refinement relationships for the software prototype.

IFC Overview [1] is a webpage that include short summary about IFC and links to their releases of specification. IFC4 is the format that is referred to in this document.

RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax [6] is a specification for URI which are used in IFC4's IfcURIReference. Linking syntax is crucial for referring to external objects as the application need to parse the link correctly and in the same way.

3 Solutions

To have relationships or links between multiple levels of development, there could be many ways to do this.

1. To have single IFC file containing multiple LOD.
2. Each IFC file contain relationships or links to other IFC file.
3. To put all IFC files from different LOD and another file containing relationships into a single zip file.

3.1 To have single IFC file containing multiple LOD.

For the first approach, to have single IFC file containing multiple LOD, this means that multiple LOD are represented in a single .ifc or .ifcxml file, this will need a lot of modification of IFC specification and entities. The structure needs to allow one file to have multiple LOD. The applications which uses current versions of IFC format are also needed to be changed and adapted greatly to support new specification. To be able to design schema for this single IFC file, extensive knowledge and experience of multiple LOD construction is required to design optimally and cover all use cases so I did not tackle this approach to design complete specification, just only concepts. Three important things that the single IFC file have to contain are all the objects in each LOD, relationships inside each LOD, relationships between LODs.

3.2 Each IFC file contain relationships or links to other IFC file.

For the second approach, each IFC file contain relationships or links to other IFC file, this means that each file from different LOD file will have to store relationship or links to other LOD's IFC file, this will requires minor additions to the IFC specification.

As IFC are object based, structured in the way that each object will have attributes which the value could refer to a value, another entity (used for relationships). One object could be refined by multiple objects in the next LOD, so we need another relationship for storing the relation and an inverse attribute for IfcObject for that. I would name the inverse attribute "IsRefinedBy" and the relationship "IfcRelRefines". This is because the existing child class of IfcRelationship entities for IFC includes IfcRelAssigns, IfcRelAssociates, IfcRelConnects, IfcRelDeclares, IfcRelDecomposes

and IfcRelDefines. Which all of them are not suitable to be used for linking to refined objects because of their entity definition suggest it for other uses. So there must be a new relationship for linking to refined objects, “IfcRelRefines”. The name is named based on existing IFC relationships by using 3rd person in present tense. And IfcRelRefines is 1-to-many relationship so attribute linking to objects would have to be named Relating+<name of relating object> for the 1 side and Related+<name of related object> for the many side according to the specification. In this case, it should be named RelatingObject and RelatedExternalObjects. The RelatedExternalObjects would have to refer to IFC objects in another file. However, there is no entity that are made for referring IFC object in another file yet. I would make a new IFC entity named IfcExternalObject for that purpose. The word External is there to make people knows that it it referring to object in other file. So this IfcExternalObject will need to have an identifier to IFC object in another file. In this case, we will use IfcURIReference as attribute named Location, similar to the existing entity IfcExternalReference. Below is an example of the IfcExternalObject node to be used for RelatedExternalObjects of IfcRelRefines for IFC4 XML.

```
<IfcExternalObject id="i482" GlobalId="17pidTtJjBZx78E2xMvCGC"
Location="file:/C:/path/walls2.ifcxml#2ZGPwZdkr9OQYsKQvp4Hvl"/>
```

Code 1 Example of IfcExternalObject

To sum up, here is an example of IFC4 XML to be added to the IfcObject.

```
<IsRefinedBy>
   <IfcRelRefines id="i375" pos=”0” GlobalId="2VW91ju5b4hfwENio7jua2">
      <RelatedExternalObjects>
         <IfcExternalObject id="i482" GlobalId="17pidTtJjBZx78E2xMvCGC"
Location="file:/C:/path/walls2.ifcxml#2ZGPwZdkr9OQYsKQvp4Hvl" pos="0"/>
         <IfcExternalObject  id="i483" GlobalId="1tdoq_o1X73RYPsHbJXv"
Location="file:/C:/path/walls2.ifcxml#0RsRtb6QDCve2_5KS4cFlm" pos="1"/>
      </RelatedExternalObjects>
   </IfcRelRefines>
</IsRefinedBy>
```

Code 2 Example of IFC4 XML with refines relationship

In this case, this object is refined by two objects in the next level of development. If in the next level of development, the object is deleted, RelatedExtenalObjects will not contain any nodes.

For IFC4 Express specification, that would be

```
ENTITY IfcRelRefines
    SUBTYPE OF IfcRelationship;
        RelatedExternalObjects  :  SET [1:1] OF IfcExternalObject;
        RelatingObjects         :  IfcObjectDefinition;
END_ENTITY;
```

Code 3 IfcRelRefines Express specification

The disadvantage of this approach is that the links are hard to maintain. Just a file rename or move may cause the links to become invalid. In order to solve that problem, we can put all those IFC files from different LOD into an archive/zip. The location attribute of the IfcExternalObject can now be and URI referring to relative path like below.

```
<IfcExternalObject id="i482" GlobalId="17pidTtJjBZx78E2xMvCGC"
Location="./walls2.ifcxml#2ZGPwZdkr9OQYsKQvp4Hvl" pos="0"/>
```

Code 4 Example of IfcExternalObject with relative path

We will call this zip file “Linked Multi LOD IFC Zip” and use extension “.lmifc”.

3.4 To put all IFC files from different LOD and another file containing relationships into a single zip file.

For the third approach, to put all IFC files from different LOD and another file containing relationships into a single zip file, this will requires no changes to the IFC specification. The advantage is that the data for multiple LOD resides in different file from IFC files of each LOD, this makes it possible to make a specification for multiple LOD as an upper layer without interfering to semantic of IFC file that are used widely. This approach is also corresponding to concepts in [4].

The third approach have the most advantage so it is the selected as recommended solution for this document. We will call that zip file “Multi LOD IFC Zip” or in short “MIFC” file. The file extension to be used is also “.mifc”. The format of the additional file that will store multi LOD informations are needed to be designed. I have decided to use JSON over XML or STEP as a data format for this file because of the popularity, file size and availability of tools and libraries. What every file multi LOD zip file should have is the sequence of LOD, IFC file format and version. I would name that file “metadata_v1.json”. The reason I have added version inside the file name is that one MIFC file could be able to support multiple specification versions of data at once.

Below is an example how “metadata_v1.json” could look like.

```
{
    "format": "IFC-SPF",
    "version": "4",
    "files": [
        "lod1.ifc",
        "lod2.ifc"
    ]
}
```

Code 5 Example of metadata_v1.json for IFC-SPF

If the IFC file is in XML format, it will look like this.

```
{
        "format": "IFC-XML",
        "version": "4",
        "files": [
                "lod1.ifcxml",
                "lod2.ifcxml"
        ]
}
```

Code 6 Example of metadata_v1.json for IFC-XML

For this document, we will be focusing on storing refinement relationship which describe which element in one LOD became an element in the next LOD. I would put it in separate file named “refinements_v1.json”. The reason I do not put refinement relationships inside “metadata_v1.json” is that if we have multiple aspect of multi LOD in separate file, it will be easy to extend by adding new file every time we want to add new aspect. The specification for each aspect then can be developed independently. The application that can read “refinements_v1.json” will not be effect if we add other files. Here is how “refinements_v1.json” could look like.

```
{
        1: {
                "1tdoq_o1X73RYPqkHJ1542": [
                    "1tdoq_o1X73RYPqkHJ1242",
                    "1tdoq_o1X73RYPqkHJ1678"
                  ],
                "1tdoq_o1X73RYPqkHJ1543": ["1tdoq_o1X73RYPqkHJ1245"],
                "1tdoq_o1X73RYPqkHJ1847": []
        }
}
```

Code 7 Example of refinements_v1.json

What this means is that the first file described in “metadata-v1.json” contains three refinements. Object with GUID ends with 1542 has become two objects with GUID ends with 1242 and 1678. Object with GUID ends with 1543 has become object with GUID ends with 1245. Object with GUID ends with 1847 got deleted in the next LOD.

So the MIFC file will include IFC files for every LODs, metadata_v1.json and refinements_v1.json.

3.5 Comparing Solutions

Solution	Specification Changes	Coupling	Extensibility	Easy to Interpret
Single Multi LOD IFC File	High	High	Low	Medium
Linked Multi LOD IFC Zip	Medium	Medium	Medium	Easy
Multi LOD IFC Zip	None	Low	High	Medium

Table 1 Comparison of Solutions

To sum up, above is a comparison table describing the three solutions. Amount of specification changes needed is already described for each solution in the previous section. Coupling is high when main IFC specification has to be changed when we want to extend or add multi LOD models. With high coupling, extensibility will be hard because little changes will need to update many parts of the specification. Single Multi LOD IFC File is hard to interpret as the one who reads this IFC file would have to know many aspects of the IFC in order to parse it correctly. But the good thing is that it is in single file so that it could be parse using STEP parser or XML parser easily in one step, so I marked it as medium to interpret. The Linked Multi LOD IFC Zip is easy to interpret as programs that can read IFC4 could read the IFC file inside the zip just like before, with just only new elements to be interpret. For Multi LOD IFC Zip, it is medium to interpret because the software would need to interpret the new JSON files in addition to IFC files and recognize the relationship between them.

The third solution, Multi LOD IFC Zip is the proposed solution as it have the most advantages in long term.

4 Software Prototype

A Java software prototype was developed for demonstration. It is made to support IFC-XML format instead of IFC-SPF as it is easier to read and interpret. The program works by accepting two IFC-XML files and do geometry intersection to determine refinements. The refinements can be view and also visualize in 3D side by side. The user can also export as Linked Multi LOD IFC Zip or Multi LOD IFC Zip. The parsing of IFC-XML is done by using libraries from opensourceBIM which are in Java. Then the model is used for doing geometry intersection. At first, I have doing the software for IFC-XML 2x3, and the opensourceBIM parser works fine. However, I decided to upgrade to IFC-XML 4 and the opensourceBIM parser obviously have wrong implementation for IFC-XML 4 so it cannot parse IFC-XML 4. But for IFC-SPF 4, it can parse perfectly. So to be able to parse IFC-XML 4, I first convert IFC-XML 4 to IFC-SPF 4 using xBim libraries and it worked perfectly. For visualizing, xBIM WeXplorer javascript library was used. It is a web viewer for BIM, so the browser is needed to display. I have used Java Chromium Embedded Framework (JCEF) for embedding chromium browser in Java window. xBIM WeXplorer needs the file format .wexBIM to render so conversion is needed before rendering with xBIM WeXplorer. xBim libraries are needed for converting IFC files to wexBIM, but there is only xBim libraries in C# so the converter is written in C#. When rendering the model with xBIM WeXplorer, the GUID of selected object and refined object are also passed to the browser to highlight. For saving Linked Multi LOD IFC Zip, Java built-in library are used for manipulating XML file to include refinements as IFC parser does not support custom-made objects. For saving Multi LOD IFC Zip, json-simple library is used for creating JSON objects.

4.1 Libraries Used

- opensourceBIM pluginbase
- opensourceBIM ifcplugins
- opensourceBIM ifcopenshellplugin
- json-simple
- Java Chromium Embedded Framework (JCEF)
- Consistency Checker from [5] (Modified to support IFC4)

4.2 Usage

To use the software, at first, you need to select two level of development of your BIM file in IFC-XML format then click “Process”.

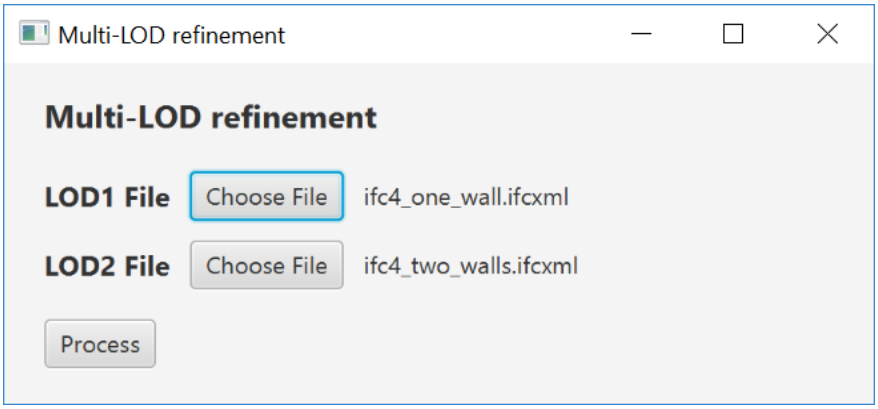

Image 1 User Interface for selecting LOD file

You will be shown with the result window. On the left side are the objects in “LOD1 File”. If you click on it, it will show refined objects in the right side.

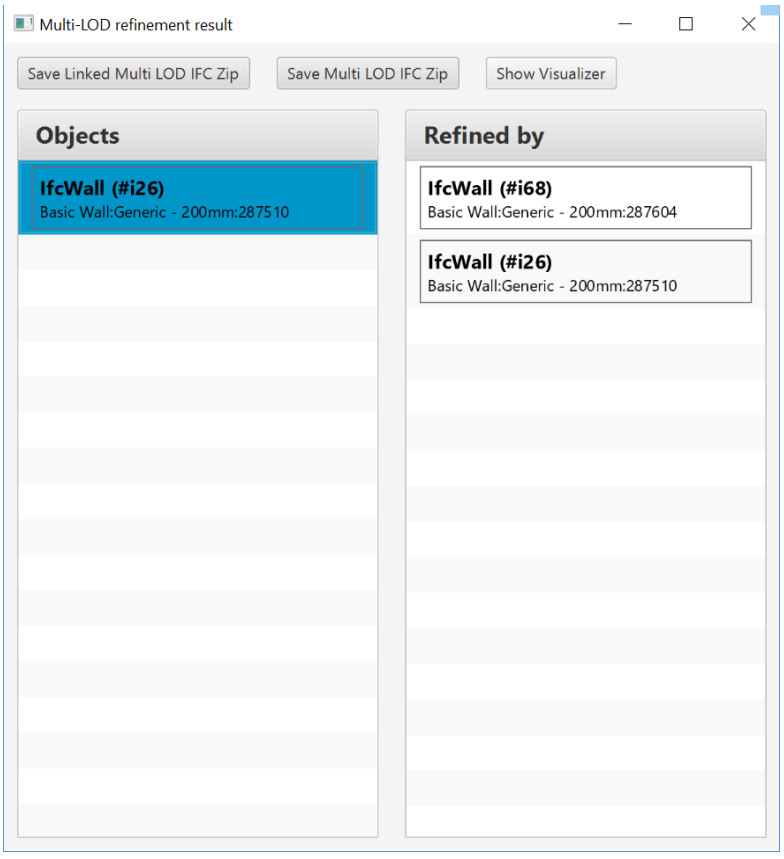

Image 2 User interface for viewing refinements

You can click on “Show Visualizer” to see the changes visually. When you click on it, a popup will appear with LOD1 displayed on the left side and LOD2 on the right side.

The selected item is focused and highlighted in red for LOD1. For LOD2, there could be many objects that are refined from selected LOD1 object and the highlighted objects must be distinguishable for different object, this is done by having the highlight for different object blink between red and green color in different interval. For this two LOD, one wall in “LOD1 File” became two walls in the “LOD2 File”.

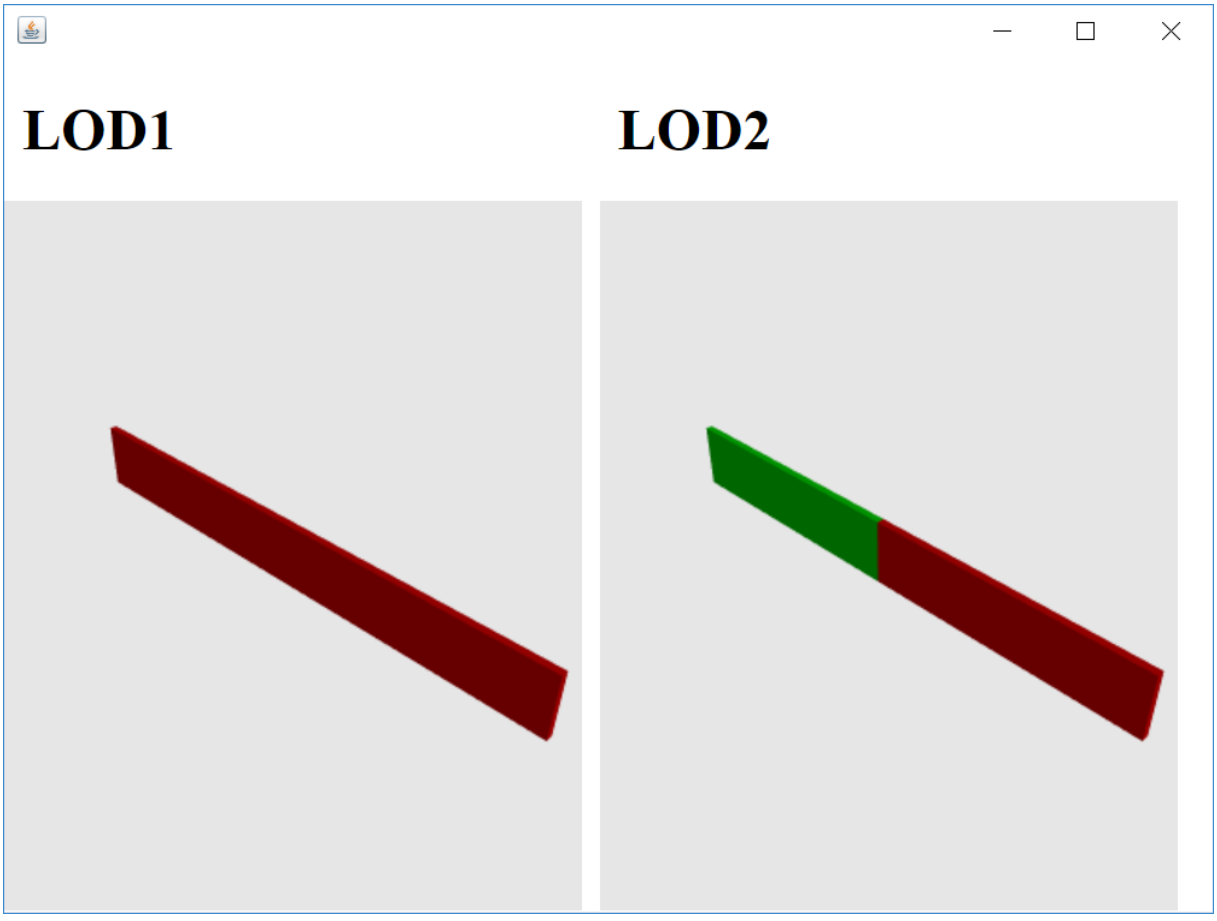

Image 3 User interface for visualizing refinements

You can save the Linked Multi LOD IFC Zip by clicking on “Save Linked Multi LOD IFC Zip”. This is done according to the solution (approach number 2). A save dialog will be shown and asked for the destination of the file to be saved.

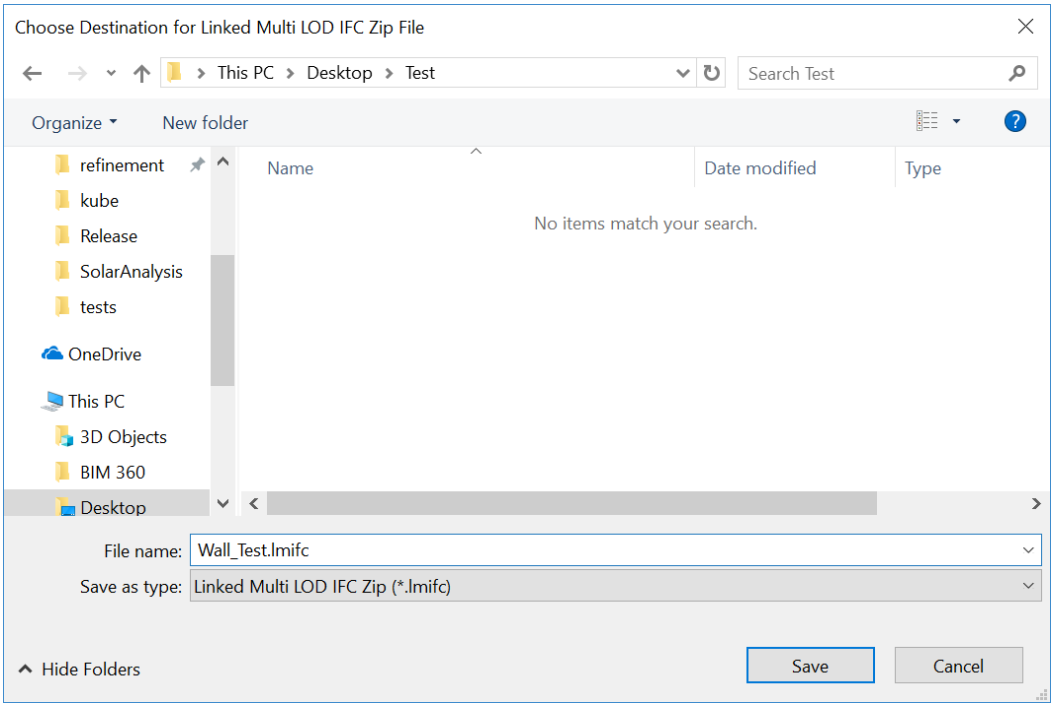

Image 4 Export Linked Multi LOD IFC Zip

After saving, if you open the file with zip viewer, you will see something like this.

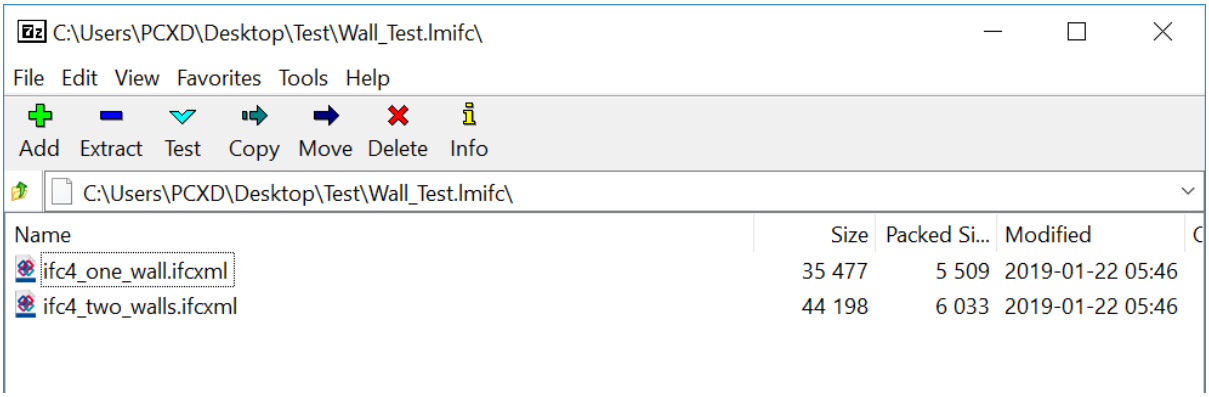

Image 5 Content of Linked Multi LOD IFC Zip

The LOD1 file, “ifc4_one_wall.ifcxml” will now contain refinement links from IfcProduct to IfcProduct in “ifc4_two_walls.ifcxml”. Here is how new IfcWall would look like.

```
<IfcWall GlobalId="1tdoq_o1X73RYPqkHJXv$H" Name="Basic Wall:Generic - 200mm:287510" ...>
  <OwnerHistory .../>
  <IsTypedBy .../>
  <IsDefinedBy .../>
  <ObjectPlacement .../>
  <Representation .../>
  <IsRefinedBy>
    <IfcRelRefines GlobalId="2GjPRF_VL51PMemShM8Hnn" id="i370" pos="0">
      <RelatedExternalObjects>
        <IfcExternalObject Location="./ifc4_two_walls.ifcxml#1tdoq_o1X73RYPqkHJXv_p" id="i368" pos="0"/>
        <IfcExternalObject Location="./ifc4_two_walls.ifcxml#1tdoq_o1X73RYPqkHJXv$H" id="i369" pos="1"/>
      </RelatedExternalObjects>
    </IfcRelRefines>
  </IsRefinedBy>
</IfcWall>
```

Code 8 Example IfcWall with refines relationship

So this wall became two walls in “ifc4_two_walls.ifcxml”.

You also have option to save Multi LOD IFC Zip file by clicking on “Save Multi LOD IFC Zip”. This is done according to the solution (approach number 3). A save dialog will be shown and asked for the destination of the file to be saved.

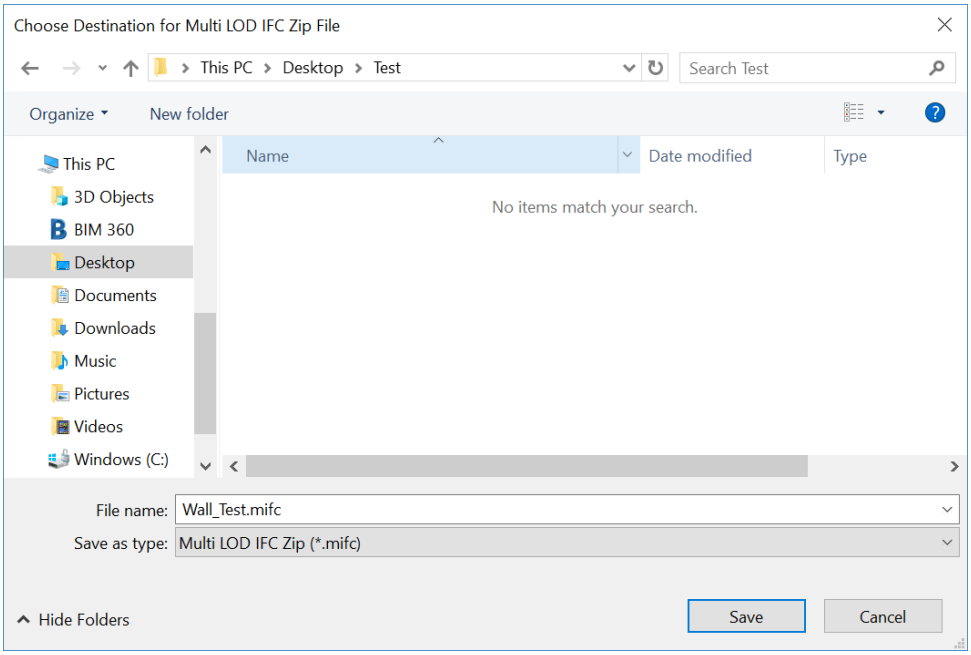

Image 6 Export Multi LOD IFC Zip

After saving, if you open the file with zip viewer, you will see something like this.

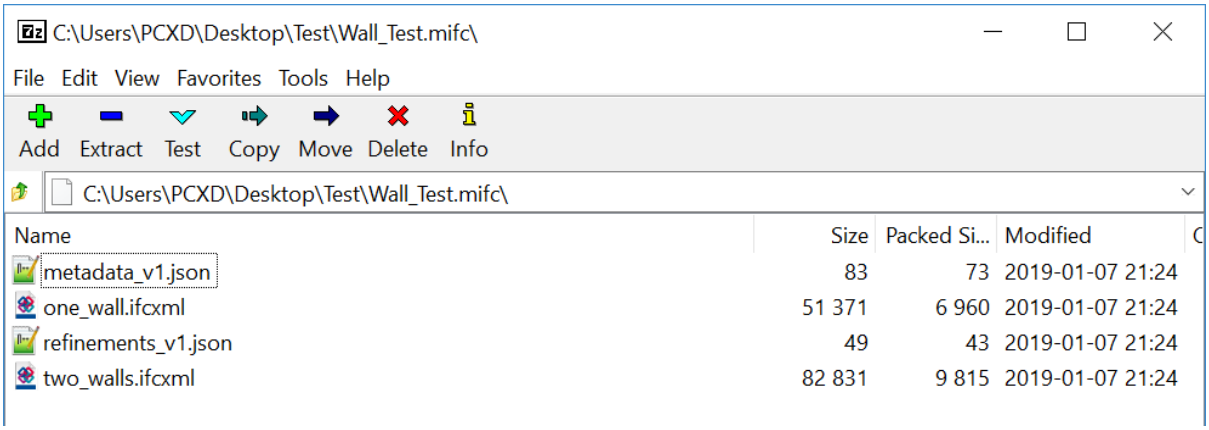

Image 7 Content of Multi LOD IFC Zip

The content of “metadata_v1.json” will look like this.

{"format":"IFC-XML","files":["one_wall.ifcxml","two_walls.ifcxml"],"version":"4"}

Code 9 Example of metadata_v1.json

The content of “refinements_v1.json” will look like this.

{"1":{"i218":["i1854"],"i267":["i1903","i1973"]}}

Code 10 Example of refinements_v1.json

5 Conclusion and Outlook

This document shows ways to link models to it refined models in the next level of development for IFC-SPF and IFC-XML. Software prototype are also developed for demonstration. For the proposed solution, Multi LOD IFC Zip, the benefit of this design is that it does not require any changes to the existing IFC specification. It is also greatly extensible. However, if new non-backward compatible version is to be developed, it would be good to also consider having all the levels of development in a single file as IFC is used for data exchange between different applications. So that the multiple LOD could be passed on easily and there would be a possibility to reduce file size by removing redundancy model, similar idea to [3].

And to be able to use SPARQL to query information from the model, model in RDF format is needed. Currently, there are only tools for converting IFC-SPF file to RDF format. Another software for converting multi LOD zip file to RDF will needed to be developed.

For the software prototype, we can see that parsing using libraries are still not easy as it should be. Maintaining IFC specification is important. But I think having great parsing libraries supports in variety of languages is also crucial. With specification on software interface for parsing and libraries support in multiple languages, the newer IFC format could be more adopted.

Bibliography

[1] "IFC Overview," buildingSMART, [Online]. Available: http://www.buildingsmart-tech.org/specifications/ifc-overview. [Accessed 24 April 2019].

[2] M. Fischer and C. Kam, "PM4D final report," CIFE, Stanford University, 2002.

[3] F. Biljecki, H. Ledoux and J. Stoter, "Improving the Consistency of Multi-LOD CityGML Datasets by Removing Redundancy," in *The Selected Papers of the 3D GeoInfo 2014*, Springer, 2015, pp. 1-17.

[4] S. Fuchs and R. J. Scherer, "Multimodels - Instant nD-Modeling using original Data," *Automation in Construction,* vol. 75, 2016.

[5] J. Abualdenien and A. Borrmann, "A meta-model approach for formal specification and consistent management of multi-LOD building models," *Advanced Engineering Informatics,* vol. 40, pp. 135-153, 2019.

[6] "RFC 3986 - Uniform Resource Identifier (URI): Generic Syntax," IETF, [Online]. Available: https://tools.ietf.org/html/rfc3986. [Accessed 24 April 2019].