ifcXMLNetwork: A Visual Interface for Exploring and Understanding ifcXML Data

Nayantara Duttachoudhury*
University of British Columbia

ABSTRACT

IfcXML files are domain specific type of XML files which are generated from building information models (BIM). Unlike most XML files, they do not have an inherent hierarchy of their own. Data instances in the ifcXML file are identified through unique identifiers and these are used to connect one data instance to other. This results in long chains of connections through multiple reference identifiers. At present there is not effective method of extracting and understanding these connections. The only way a user can see how one data instance is connected to another is by manually searching and following this path of connections through the ifcXML file.

This paper introduces a new system called **ifcXMLNetwork** which generates a network for each data instance by finding all the other data instances it is connected through its identifier. The purpose of ifcXMLNetwork is to make it easier for users to find relationships and connections in ifcXML files without manually navigating through the data file.

1 INTRODUCTION

Building Information Modelling (BIM) [3] is a process involving the generation and management of digital representations of physical and functional characteristics of places. Building information models (BIMs) are files which can be exchanged or networked to support decision-making about a place. Current BIM software is used by individuals, businesses and government agencies who plan, design, construct, operate and maintain diverse physical infrastructures, such as water, wastewater, electricity, gas, refuse and communication utilities, roads, bridges and ports, houses, apartments, schools and shops, offices, factories, warehouses and prisons. Autodesk Revit (shown in fig 2) [2] is one such BIM software. BIM models created in Revit can be saved in IFC [4]form.

IFC (Industry Foundation Classes) is a data model developed by the International Alliance for Interoperability and is used for modelling in the building industry. IfcXML [5] is its XML specification. IFC files can downloaded from building models created in Autodesk Revit [2]. Autodesk Revit is a Building information modelling (BIM) software for architects, structural engineers, MEP engineers, designers and contractors. IFC/ifcXML specifications based convertors can be used to convert IFC to ifcXML. Besides IFC, there are other standards that building data can be exported into such as Microsoft Access, gbXML or DWL. But it is seen that ifcXML contains information not supported by other standards, so it is preferred. There are many challenges with ifcXML data. Compared to other standards, it is big and has the most complex schema [1].

ifcXML files are a domain specific type of XML files with a complex schema structure. But they do not have an inherent hierarchy of their own. Data instances in the ifcXML are identified through unique identifiers knows as **id** and connected to each other through reference identifiers known as **idRef**. This results in long chains of connections across multiple id's. The only way a user can see how one data instance is connected to another is by manually searching and following this path of connections through the ifcXML file.

ifcXMLNetwork aims to make this process much simpler by generating a network for each data instance by finding all the other data instances it is connected through its id. Users can now easily find relationships and connections in ifcXML files without navigating through the data file.

*e-mail: nduttac@cs.ubc.ca

2 RELATED WORK

XML visualization is not a new concept. There are many online XML viewers [7] that extract the hierarchical form of XML data and visualize it as a tree. But each XML file is different from another. XML files across different domains are very different. Domain specific visualization of XML data has been done. These visualizations are distinct from each other, even though they all have XML data in common. This shows that XML data visualization is very domain specific.

2.1 Linguistic XML

In XCES [9], the authors have created an XML-based standard encoding for linguistic corpora. This shows us that the need for standardizing XML data is consistent throughout all domains. Dipper et al. [8] have applied visualization to linguistic XML data to find patterns in annotated data. After that, an OWL and XQuery based mechanism [10] was used to retrieve linguistic patterns from XML-Corpora. Finally in 2008, an ontology of linguistic annotations [11] was created based on existing standardizations for integration of linguistic data in XML form.

2.2 Genomic Visualization of XML Data

CGView [12] is a Java application and library for generating high-quality, zoomable maps of circular genome. It converts XML data to a graphical map. In some cases, XML languages such as PhyloXML [13] were created to store and exchange the structures of evolutionary trees and associated data from the complex schema described through an XSD. PhyloXML was extended to a visualization known as Interactive Tree of Life [14] by Ivica Letunic et al.

2.3 XML Data visualization Software

GGobi [16] is an interactive and dynamic for data visualization. One of the data types it visualizes is XML data. Software architectures based on XML data have also been visualized [15]. XML-based static type checking and dynamic visualization for TCOZ [17] develops a type checker for detecting static semantic errors in the TCOZ specification. There are many more XML data visualization softwares, but they all cater to different kinds of XML data.

2.4 IFC Visualization

Visualization of ifcXML data is a field that remains unexplored. There are a few IFC checkers and viewers in the market. IFC check-

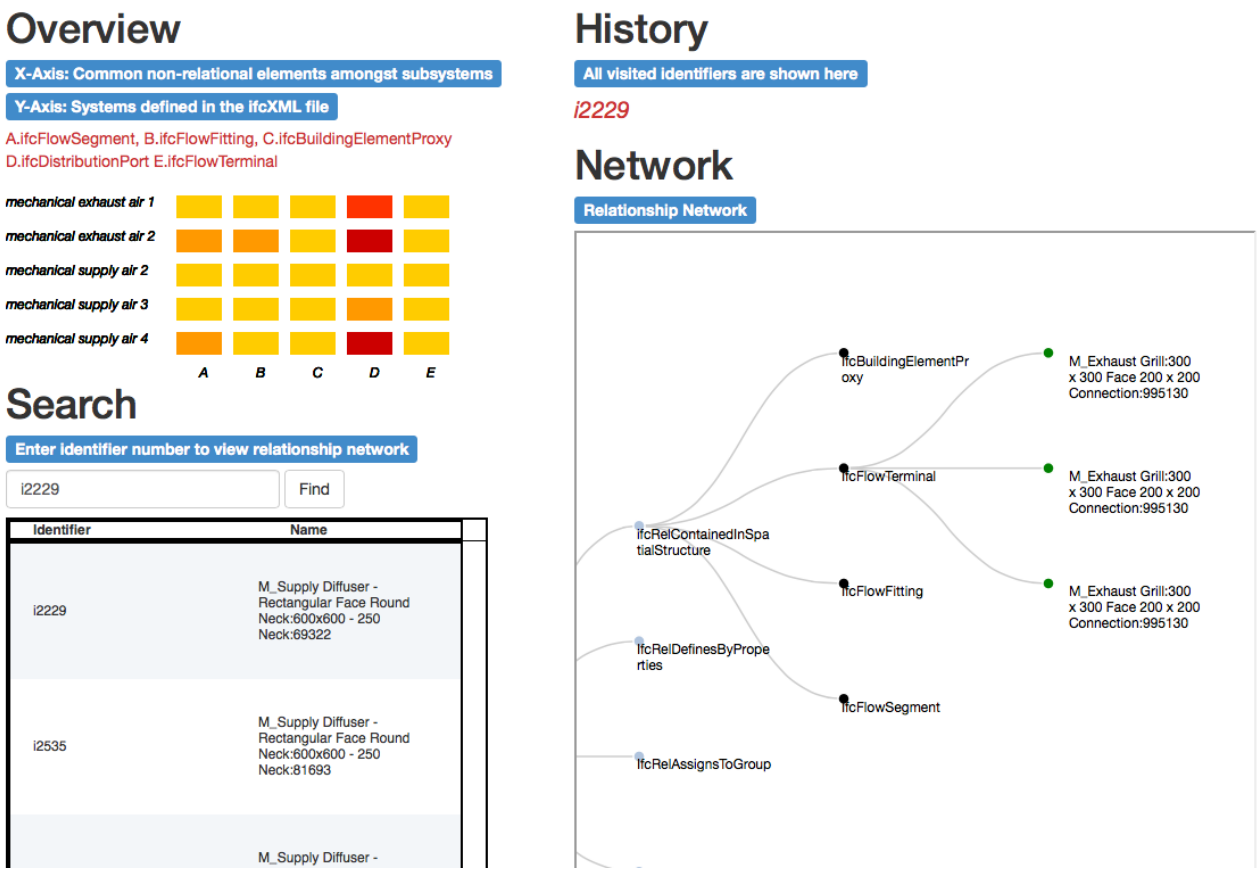

Figure 1: Screenshot of complete System.

ers check if an IFC file is defined correctly and all IFC files related to the same project are appropriately integrated. State of the art IFC viewers like Solibri extract information from IFC files and display a 3D model of the building. These viewers make it possible for a user to navigate through the virtual building and explore its spatial structure. Information such as dimensions of a flow terminal and other spatial details about objects in a building can be found through Solibri (fig 3). But it fails to answer many questions whose answers are deeply embedded in the data and not directly related through spatial details. Understanding how different objects are related to each other and by what relationship is one such situation. Also since Solibri has a nomenclature of its own, there are data inconsistencies. In such cases, the only option is to explore the raw ifcXML data.

3 Problem Description

IfcXML files have very complex schemas. Most XML files have an underlying tree hierarchy to them. In ifcXML files, this hierarchy is redundant. The entire file is defined in the root node. Following the root node, all the possible different elements with their data instances are defined as a list of 2nd level nodes. This can be seen in fig 4 where 'ifc::ous' is the root node under which the entire file is defined. Following that is a list of elements such as *ifcRelConnectsPathElements*.

We can understand the schema by drawing an analogy to relational databases, even though ifcXML data is unstructured. ifcXML is a domain specific type of XML data. Like tables in relational databases, we have elements in ifcXML. These are of two types, relational and non-relational. Non-relational elements give new information about an object and relational elements connect two non-relational elements through a relationship. For example, *ifcContainedinSpatialStructure* is a relational element connecting two data instances in non-relational elements such as *ifcFlowTerminal* and *ifcDistributionPort*. This connection tells us that a data instance in *ifcFlowTerminal* is in the same spatial structure as another data instance in *ifcDistributionPort*. In fig 5, A and C are non-relational elements while B is a relational element.

Each element has a certain number of data instances in it, like tables have rows. Each element has attributes, like names of columns in tables which have separate values for every row. One attribute is reserved for the id, which is the global identifier for each data instance. Many data instances also have an idRef. This acts as a pointer to another data instance in a different element. In relational databases, we have foreign keys in tables which point to primary keys of other tables. For ifcXML, instead of pointers between elements, idRef's are pointers between specific data instances. Properties of objects are often not attached directly to a single element but related indirectly through these reference identifiers. Some of these reference paths are very long. Analyzing how different elements and their attributes are linked to get information about objects and their properties is the biggest challenge with ifcXML data.

The purpose of ifcXMLNetwork is to help understand ifcXML data better by visualizing it. The dataset used is the ifcXML file of the mechanical model of the CIRS building in UBC. Here is a recap of the terminology to understand the data better.

3.1 Terminology

- Schema: Structure of ifcXML file.

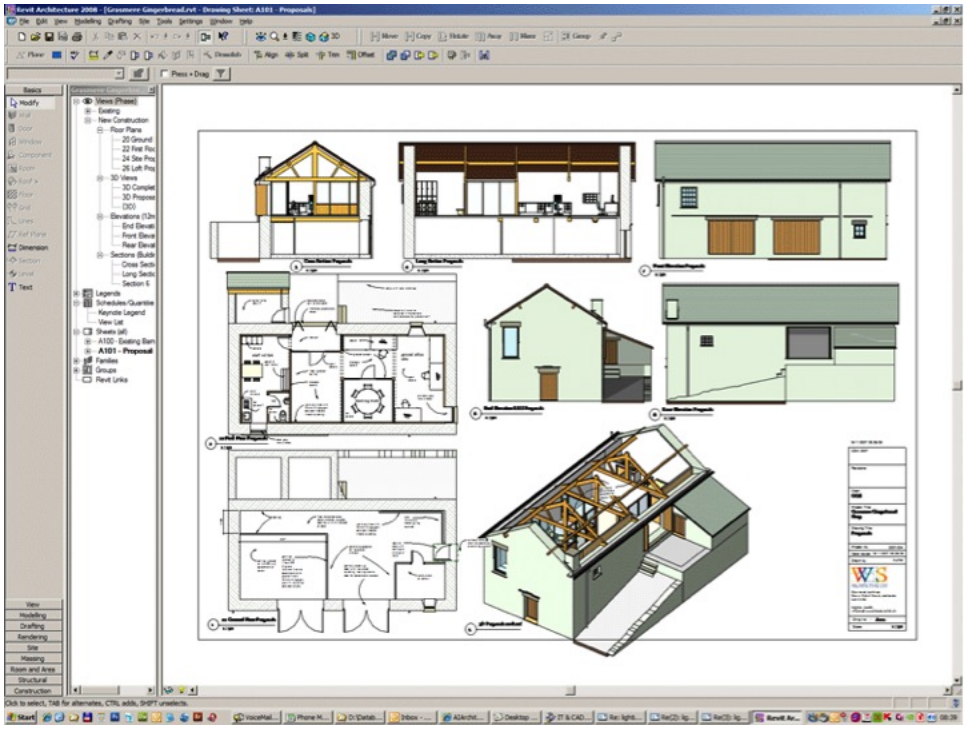

Figure 2: Building Information Models in Autodesk Revit

- Objects: Real world objects in the building. E.g. flow terminal in room 007 in the basement.
- Properties: Information about objects. E.g. type of flow terminal.
- Elements: Similar to tables in relational databases. Are of 2 types - relational and non-relational. Relational elements connect non-relational elements to each other through id's.
- Data instances: Rows of data in tables.
- Attributes: Columns in a table with separate value for each data instance or row.

3.2 Usage Scenario

Rob, a civil engineer has the IFC (or ifcXML) file of the CIRS building in UBC. He wants to see which distribution ports belong to the same system as a particular flow terminal. He tries to convert the IFC file to a .rvt file to view in Revit. But this transformation leads to loss of information. Next he uses an IFC viewer like Solibri. But Solibri has naming conversions of its own. This leads to inconsistencies in the data. Also, Solibri cannot capture all the information from the IFC file. Now, the only solution for Rob is to manually extract information from the ifcXML data. But the ifcXML is very huge with information spread across the file. Our visualization aims to make this process much easier for Rob.

4 Design

The goal of the project is to create a simple and easy-to-use visualization that shows the different relationship between elements in an ifcXML file. It allows users to see how different data instances in non-relational elements are connected through relational elements. For example, a user might need to know which distribution ports are connected to the same flow terminal. Our system makes it easy to find relationships like these. All a user would need to do is generate the relationship network of the particular flow terminal and check the distribution ports connected through the relational element *ifcRelConnectsPortToElement*.

4.1 Tasks

Our visualization system helps the user perform these tasks:

- **Summarize** all common non-relational elements in terms of the subsystems they belong to.
- **Locate** a data instance whose identifier is already known.
- **Explore** and find a particular data instance from the tables generated from the different segments in the overview view.
- View the relationship **network** of a particular data instance to see which other data instances it is connected to and through what relationship.

4.2 Encoding

The tasks are encoded using two main visualization idioms. The overview view is visualized using heat map where each segment is colour coded. The colour gradient goes from light yellow to dark red as seen in fig 7. The range of items is 0 - 100 with each coloured section for 20 items. Node-link diagrams are used to encode the relationship network where red and green nodes refer to data instances, blue nodes refer to the relational element and black nodes refer to non-relational elements. The data is manipulated by selecting sections in different views.

5 Solution

Our visualization is divided into four views. Out of these, two views capture the main essence of the project. The first thing a user sees is the overview view which shows common non-relational elements and their corresponding data instances through different systems.

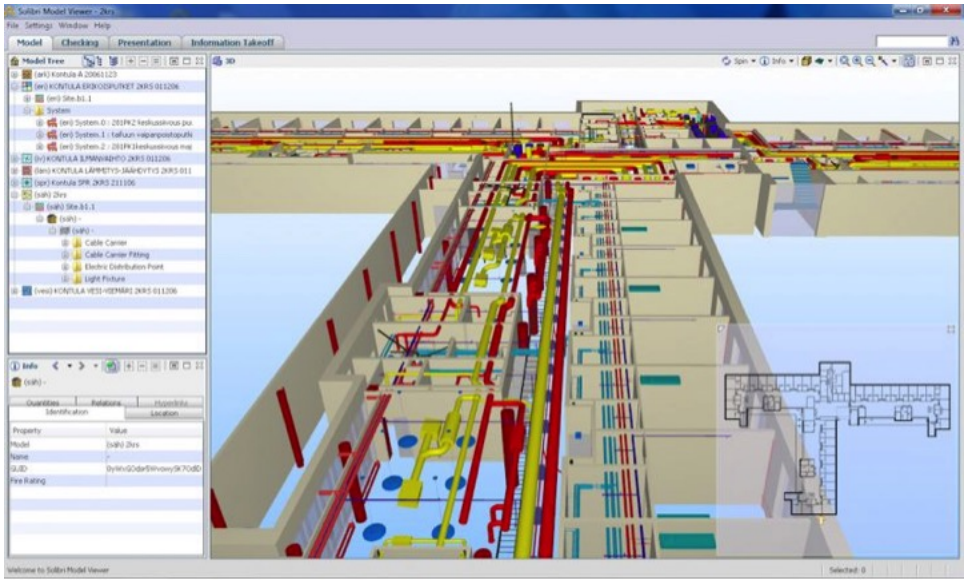

Figure 3: Screenshot of Solibri

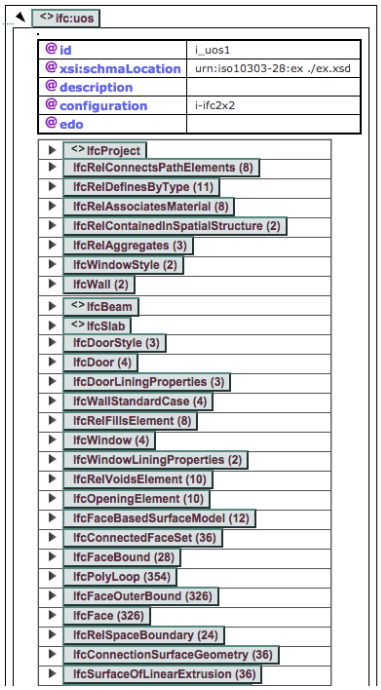

Figure 4: Structure of ifcXML Schema

Each segment in this view has its own set of data instances in tabular form. This table is generated in the search view along with a search bar. Putting the value of an identifier in the search bar displays the graph of the data instance the identifier refers to. This graph is known as a relationship network and is one of the most important parts of the visualization. The relationship network displays how the selected data instance is connected to other data instances in non-relational elements through relationships described by relational elements. The analysis table (fig 12) of the visualization gives a detailed analysis of the data, the tasks involved and the encodings done.

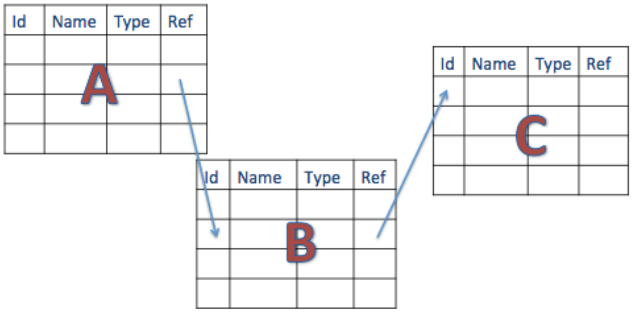

Figure 5: Relational and Non-relational Elements

5.1 Overview

IfcXML files have a special non-relational element type called *ifcSystem* defined in them. Data instances in *ifcSystem* are defined through attributes such as Name and Type. All data instances in *ifcSystem* can be classified into system types such as Mechanical Exhaust Air. Our dataset has 2 types of Systems: Exhaust Air and Supply Air. These systems are divided into 5 subsystems: Mechanical Exhaust Air 1, Mechanical Exhaust Air 2, Mechanical Supply Air 2, Mechanical Supply Air 3, Mechanical Supply Air 4.

Data instances of other non-relational elements such as *ifcDistributionPort* are connected to particular individual data instances in *ifcSystem* through the relational element *ifcRelAssignsToGroup*. Each data instance in *ifcSystem* is connected to multiple data instances of a set of non-relational elements through the relational element *ifcRelAssignsToGroup* .Non-relational elements such as *ifcFlowSegment*, *ifcFlowFitting*, *ifcBuildingElementProxy*, *ifcDistributionPort* and *ifcFlowTerminal* are common in all subsystems.

The overview categories these subsystems according to the common non-relational elements. The Y axis shows the different data instances of the element *ifcSystem* defined in the ifcXML file. The X axis shows all common non-relational elements. Each rectangu-

Figure 6: Different data instances connected through reference paths.

Figure 7: Colour Gradient

lar segment in the view is colour coded according to the number of data instances. Fig 8 shows the overview view of the ifcXML file. Hovering over the rectangles in the view show the number of data instances that fall under that category. For example, hovering over the rectangular segment defined by Mechanical Exhaust Air 2 on the Y-axis and B on the X-axis shows a tooltip with the number 21 on it. This is the number of data instances belonging to this segment.

5.2 Network

The network view shows the user the relationship network of the data instances selected by it's unique identifier. The root node shows the selected data instance in a non-relational element. Second level nodes show the different relational elements that the root node refers to by it's idRef. Third level nodes show the non-relational elements connected to the relational elements in the second level. Finally, leaf nodes show the data instances connected to the root node through idRef's across the second level and third level nodes. It is seen in fig 9, the root node is coloured in red and specifies the identifier whose relationship network is being shown. Relational elements are shown as blue nodes, non-relational elements as black nodes and leaf nodes which refer to data instances connected to the root node are shown in green. Hovering over the green leaf nodes will show the unique identifier of the data instance.

5.3 Search

The Search view consists of the search bar and a table of data instances. When a segment in the overview is clicked, a table of data instances belonging to that segment is generated. The table shows

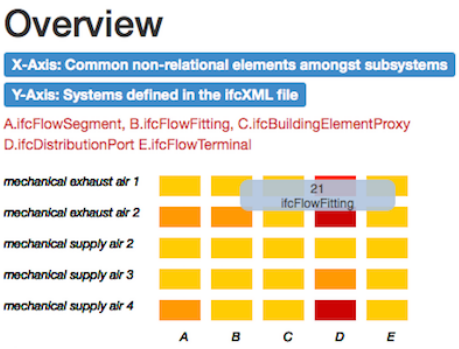

Figure 8: Overview View

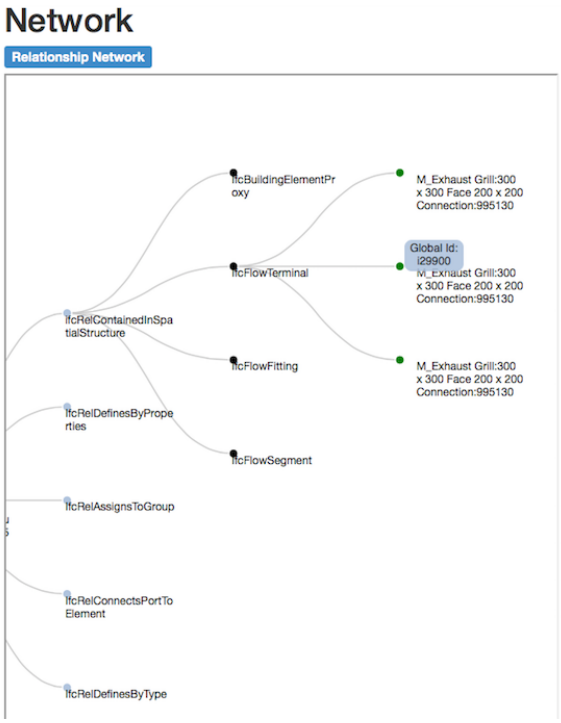

Figure 9: Network View

their id's and names. An id can be selected from this table and entered into the search box. This generates the network view of the selected data instance. A known id can also be directly inputted into the search box without finding it from the table to get it's network view. The search box and table is shown clearly in fig 10.

5.4 History

The history view gives a list of all identifiers whose networks have been generated. Identifiers next to each other refer to data instances which are connected to each other through a relational element. It also helps the user keep track of all visited data instances. From fig 11, we see that identifiers 'i2229' and 'i140134' are connected to each other through a relationship defined by a relational element.

6 Implementation

The main purpose of this project is to help domain experts extract relationship information from ifcXML files. The system needs to be easy to install and run. We chose to design a browser-based

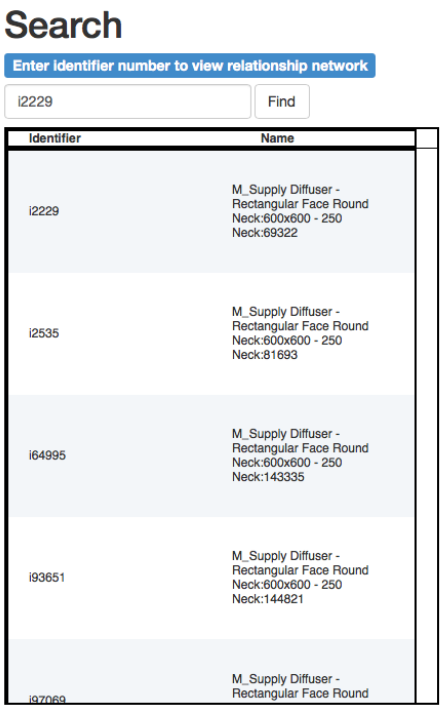

Figure 10: Search View

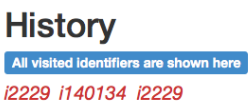

Figure 11: History View

system that just requires the users to run the .html file. The system does not need any additional downloads or installations. The required javascript libraries have been added into the source code folder. Code for extraction of data from ifcXML files are included in the project directory too, along with sample parsed data.

6.1 Data

Our original data is in the form of ifcXML files with real world objects described in terms of elements with different attributes. Each of these elements have multiple data instances identified with unique id's with values for each attribute. The data derived from these ifcXML files are as follows:

- Matrix for overview where x axis shows non-relational elements common in all subsystems and y axis shows subsystems defined in ifcXML file in *ifcSystem*.
- Each segment in the overview view has a table of data instances identified through unique identifiers.
- Each data instance has it's corresponding relationship network saved as a .json file. This file is read by a Javascript file, written using D3js which converts the JSON data into a relationship network.

6.2 Tools Used

The visualization is browser based and developed using a combination of HTML, CSS and Javascript. D3.js cite is Javascript library that uses digital data to drive the creation and control of dynamic and interactive graphical forms which run on web browsers cite wiki. Bootstrap, a web framework was used to create the framework of the visualization. The backend data processing is done using python. Python has many string processing functions which makes it an optimum choice to retrieve necessary information from the ifcXML files.

6.3 Code

The high level structure of the code is as follows:
Project Directory
- **index.html**
- **index-style.css**
- onclick.js (separate Javascript function)
- Sample JSON and HTML network data
- d3-tablesort library files

d3 library files
Bootstrap files
Backend data processed in python with python code.

The main code is in **index.html** which uses the stylesheet **index-style.css**. The framework of the visualization is created in this file. This file also contains the code for the overview, history view and search bar. The network view for a unique identifier is called from this file. The table for each segment in the overview is defined in the d3-tablesort files. The onclick( ) function calls the d3-tablesort function to generate the required table for the segment clicked on the overview.

7 Results

The system visualizes relationship data from ifcXML files. It opens up with the overview view and the search bar. If the user knows the identifier of the data instance he is interested in, he can directly input it into the system to generate its relationship network. Otherwise, the user can use the overview view as a starting point. The overview view shows all the systems defined in the ifcXML file through the non-relational element - *ifcSystem*. Each system has a set of common non-relational elements with a varying number of data instances. Hovering over each of the rectangular segments in the overview gives the user the number of data instances in that segment. These segments are color coded to show at a glance the number of data instances it contains. Clicking on any segment displays a table under the search bar which shows the identifiers and names of all the data instances in that segment. Entering an identifier in the search bar displays its relationship network. This network can be explored to see all the non-relational elements and data instances it is connected to through different relational elements. The history view keeps track of the identifiers visited by the user and consecutive identifiers refer to data instances connected through a relationship in the ifcXML file.

Let us take Rob again and discuss how this visualization system makes it much easier for him to retrieve the information he needs. Assuming he knows the id of the data instances that refers to the flow terminal he is interested in, he inputs that number into the search bar. Clicking on 'find' generates the relationship network of this data instance. The second level blue nodes show all the relationships the flow terminal is a part of. The relational element *ifcSystem* tells him all the other non-relational elements that are part of the same system as the flow terminal. He clicks on this node and it shows him all the non-relational elements. Since he is only interested in the distribution ports, clicking on the node *ifcDistributionNode*, will show him a list of names of all the distribution ports that belong to the same system as the flow terminal. Hovering over the nodes will give him their id's.

The visualization was shown to civil engineers who often work with ifcXML data. They were very excited with the project and

What : Data	ifcXML files (Domain specific xml files) with real world objects described in terms of elements with different attributes. Each of these elements has multiple unique data instances with values for each attribute.
What: Derived	• Matrix for heatmap where x axis shows non-relational elements common between all subsystems, and y axis shows systems (divided into subsystems) defined in the ifcXML file. • Each section in the heatmap has a table of data instances (identified through a global identifier). • Each data instance creates its own node-link diagram in form of a tree where the root node is the selected data instance, the leaf nodes are the other data instances it is connected to through reference identifiers. The middle level nodes describe the relationship between the two nodes.
Why: Tasks	Actions: summarize and identify from heatmaps, search (locate and explore) Target: Network topology (relationship between root and leaf nodes)
How: Encode	Heatmaps, Node-link diagrams
How: Manipulate	Select (elements between different views)
How: Facet	Juxtapose linked views – overview & small multiples
How: Reduce	Item Filtering
Scale	Data instances : Ten thousands All Elements: 50, Subsystems: 5

Figure 12: Analysis Table

claimed that the network view was the most helpful part of the visualization.

8 Discussion and Future Work

Our visualization shows relationships between different elements in an ifcXML file. These relationships are hard to extract from the ifcXML data as it involves going through many id's and idRef's to find elements connected to each other. The visualization is easy to use and understand. The design is pleasing to the eye and all extracted information is appropriately displayed to the user.

There are some limitations to the visualization. The history view could be improved to display the name of relationships between visited data instances. In the search view, clicking on an element in the table should be able to display it's relationship network directly without manually inputting it into the search bar. Also another view can be created which shows properties of a data instance which are defined with it's own definition. For example, 'object type' is an attribute defined with a flow terminal's definition. This information is not captured in our visualization. While making the decision to include or not include this information, domain experts told us that this information is not as interesting as it is easily available.

Currently ifcXMLNetwork only concentrates on the relationship between elements through their data instances. It would be interesting to see what other information can be extracted from ifcXML files. IFC viewers like Solibri already deal with the orientation and spacing of objects in a building, but it might be fascinating to see what else can be visualized. We understand that not all extractable information would need to be visualized.

Before the start of the project, we were not aware of the amount of time and effort it would require to parse the ifcXML files into smaller JSON files. Even though this step took a lot of time, it validated our assumption that the process of extracting relationship information from ifcXML files is indeed very difficult.

9 Conclusion

We have presented ifcXMLNetwork which generates a network for each data instance by finding all the other data instances it is connected through its identifier. The purpose of ifcXMLNetwork is to make it easier for users to find relationships and connections in ifcXML files without manually navigating through the data file. Users can start by viewing the overview and finding which non-relational elements they are interested in and under which system. A table of data instances identified by their id's is generated for every rectangular section of the overview. A selected identifier can be entered into the search bar to display its relationship network. This network can be explored to find other data instances the identifier is connected to and through what kind of relationship. A history view makes it easy for the user to keep track of all the visited data instances through their id's.

References

[1] Jiemin Zhang. Evaluations on XML Standards for Actual Applications. 2013

[2] Autodesk Revit. http://www.autodesk.com/products/revit-family/overview

[3] Building Information Model http://www.autodesk.com/solutions/building-information-modeling/overview

[4] Industry Foundation Classes http://www.buildingsmart.org/standards/ifc

[5] ifcXML http://www.buildingsmart-tech.org/specifications/ifcxml-releases

[6] Solibri http://www.solibri.com/products/solibri-model-viewer/

[7] XML Grid http://www.xml-grid.com

[8] Dipper, Stefanie, and Michael Gtze. "Accessing heterogeneous linguistic data-generic XML-based representation and flexible visualization." Proceedings of the 2nd Language and Technology Conference 2005. 2005.

[9] Ide, Nancy, Patrice Bonhomme, and Laurent Romary. "An XML-based Encoding Standard for Linguistic Corpora." Proceedings of the Second International Conference on Language Resources and Evaluation. 2000.

[10] Rehm, Georg, Richard Eckart, and Christian Chiarcos. "An OWL-and XQuery-based mechanism for the retrieval of linguistic patterns from XML-corpora." Corpus 2.3 (2007): 1.
[11] Chiarcos, Christian. "An ontology of linguistic annotations." LDV Forum. Vol. 23. No. 1. 2008.
[12] Stothard, Paul, and David S. Wishart. "Circular genome visualization and exploration using CGView." Bioinformatics 21.4 (2005): 537-539.
[13] Han, Mira V., and Christian M. Zmasek. "phyloXML: XML for evolutionary biology and comparative genomics." BMC bioinformatics 10.1 (2009): 356.
[14] Letunic, Ivica, and Peer Bork. "Interactive Tree Of Life v2: online annotation and display of phylogenetic trees made easy." Nucleic acids research (2011): gkr201.
[15] Houlding, David Ian. "Method and system for providing visualization of underlying architecture of a software system." U.S. Patent No. 7,415,697. 19 Aug. 2008.
[16] Swayne, Deborah F., et al. "GGobi: evolving from XGobi into an extensible framework for interactive data visualization."Computational Statistics and Data Analysis 43.4 (2003): 423-444.
[17] Dong, Jin Song, et al. "XML-based static type checking and dynamic visualization for TCOZ." Formal Methods and Software Engineering. Springer Berlin Heidelberg, 2002. 311-322.