Aalto University
School of Engineering

Salik Ahmed Khan

Granularity in Visualisation of 3D BIM Models

Design-Science Approach

Master's Thesis
Espoo, May 27, 2019

Supervisors: Professor Vishal Singh, Aalto University
Professor Sudhir Misra, IIT Kanpur
Advisors: D.Sc. Seppo Törmä
D.Sc. Mehmet Yalcinkaya

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of master's thesis

Author Salik Ahmed Khan

Title of thesis Granularity in Visualisation of 3D BIM Models: Design Science Approach

Master programme Building Technology **Code** ENG27

Thesis supervisor(s) Prof. Vishal Singh, Prof. Sudhir Misra

Thesis advisor(s) D.Sc. Seppo Törmä, D.Sc. Mehmet Yalcinkaya

Date 27.05.2019 **Number of pages** xii + 97 **Language** English

Abstract

Building Information Modeling (BIM) has gradually grown into one of the key information management platforms in the Architecture, Engineering and Construction industry. With the growing amount of information in and outside the digital construction industry, concepts of information retrieval like relevance and granularity have become relevant in this domain. An increased need for interoperability and easy access to information has caused the industry to look towards concepts like the semantic web and linked data. But within all this, the geometrical visualisation, which is an integral part of the BIM process, has lagged behind on the front of granularity and still seems to be done mainly using the conventional ways. We try to explore ways to introduce granularity in the visualisation of 3D BIM models, and connecting it to the semantic information which is already granular, thus creating a mapping between the two at a granular level. A web-based prototype is implemented and analysed as a proof of the presented concept, with the semantics being represented inside a graph-based data structure.

We further present a discussion on the potential applications and use cases of the conceptualised framework in the field of construction and building lifecycle management. The work aims to take the first step towards modularising the visualisation process, and has tried to pave the way for detailed analyses and further improvements that may follow in this direction.

Keywords Building Information Modeling, Granularity, Visualisation, 3D Building Models, Facility Management, Construction Management, Graph Database, Web 3D Rendering

Acknowledgements

I start in the name of Allah, the Almighty. All praises and thanks belong to Him, for His endless blessings bestowed upon me, especially during this period of my master's project. I acknowledge that my success is only from Allah. The next I would like to thank my parents for providing constant support throughout my life, always trying to provide the best for me, and allowing me to pursue what I wanted. Without your prayers and support, I would never have achieved anything which I did.

I express my gratitude to Prof. Sudhir Misra for his constant support throughout my stay at IIT Kanpur, and the process of my travel to Finland with the academic as well as the administrative issues. He is the one who introduced me to the domain of Infrastructure Management, and helped me develop an understanding and interest through his multiple courses and projects. I also thank Prof. Vishal Singh from the bottom of my heart for his acceptance to supervise my thesis as a part of the exchange program, and for his guidance and important advice throughout the project and beyond. He is a true innovator and thinker, and has over the period of this time helped me develop a systematic thinking approach towards research in general. I did not have much experience of doing research, and in the area of BIM in particular, when I first came to Finland. But through his support and guidance, I was able to accomplish a full master's project related to BIM. I would also like to thank all the other professors and instructors at IIT Kanpur who have enriched me with their teaching and guidance during the course work.

Another persons who have helped me through my project are my advisors, Mehmet and Seppo. They introduced me to the idea of granularity and its importance, and through their expert knowledge and skills in this domain, have helped me develop a more wholesome understanding of this topic. Further, they helped me tackle with the various technical and theoretical difficulties that came up during the implementation process. I would like to thank both of them wholeheartedly for their support. I would also take this opportunity to thank Daniel Zibion, who worked previously in this research group, and

whose work I built upon. He provided me with all the details of his work, explaining me how it worked and how I can contribute to it. His support has been really important for me to achieve what I did.

I found many supportive friends and colleagues during my stay at Finland, including my lab mates Siyan, Mustafa, Antti, Ala, Jerry, Jyrki, Saeed, Ana, Chris and everyone else. I would like to thank all of them for the insightful discussions we had. Those really helped me along my way, and opened my thoughts to a lot of different perspectives. Special thanks to Siyan and Antti for those badminton matches, and to Chris and Jerry for the football night. I want to specially thank brother Mustafa who has been like an elder brother all through my time in Finland, helping me out with every issue, be it academic or personal, and sincerely advising me from his experiences in life as a student, and otherwise. I found some gems of friends and neighbours in Abdullah, Omer, Taha, Waqar, Shahrukh, Jaisal, Khoa, Udit, Manoj, Kunj, Sebastian and many others who have made my stay in Finland much more wonderful and easy.

I would like to thank Aalto University and the administrative staff over here for giving and helping me with this lifetime opportunity to study as an exchange student, and for funding my stay here in Finland. I would like to thank the staff of OIR office and DOAA office, IIT Kanpur, especially Sarkar ji for his support with all the paper work and procedures. I also express my gratitude towards the DORA office, IIT Kanpur which provided me with advance funding to make my travel to Finland possible.

Finally, I would also like to mention of my friends from IIT Kanpur who have constantly been in touch with me and provided an emotional support and motivation whenever I needed. Thank you Prakhar, Prasad, Tasnim, Tau, Ayush, and all other wingies and friends for these memorable years at IIT Kanpur. I would never forget these wonderful years of my life which have transformed me into a better person than I was before.

Thank you!

Espoo, May 27, 2019

Salik Ahmed Khan

Contents

List of Tables

List of Figures

Abbreviations and Acronyms

2D	Two Dimensional
3D	Three Dimensional
ACID	Atomic, Consistent, Isolated and Durable
AEC	Architecture, Engineering and Construction
AEC/FM	AEC/Facility Management
API	Application Programming Interface
BB	Bounding Box
BIM	Building Information Modeling
BOT	Building Topology Ontology
CAD	Computer-aided Designing
CMMS	Computerised Maintenance Management System
CSG	Constructive Solid Geometry
CSS	Cascading Style Sheets
CTS	Conformance Test Suite
DCC	Digital Content Creation
DOM	Document Object Model
FM	Facility Management
GIS	Geographical Information Systems
glTF	Graphic Library Transmission Format
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HVAC	Heating, Ventilation and Air Conditioning
IAI	International Alliance for Interoperability
IDM	Information delivery manuals
IFC	Industry Foundation Classes
IfcOWL	IFC Web Ontology Language
IfcWoD	IFC Web of Data
IFMA	International Facility Management Association
IoT	Internet of Things
IR	Information Retrieval

JSON	JavaScript Object Notation
JSX	JavaScript XML
LBD	Linked Building Data
MEP	Mechanical, Electrical and Plumbing
MVD	Model View Definitions
NDC	Normalised Device Coordinates
NoSQL	Not Only SQL
npm	Node Package Manager
NURBS	Non Uniform Rational B-Spline
OWL	Web Ontology Language
PBR	Physically Based rendering
RDF	Resource Description Framework
SID	Scoped Identifier
SPARQL	Simple Protocol and RDF Query Language
SQL	Structured Query Language
STEP	Standard for the Exchange of Product model data
SVG	Scalable Vector Graphics
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Indicator
UUID	Universally Unique Identifier
W3C	World Wide Web Consortium
VARK	Visual, Aural, Read/write, and Kinesthetic
VR	Virtual Reality
WWW	World Wide Web
x3d	Extensible 3D Graphics
XML	Extensible Markup Language
XSD	XML Schema Definition

Chapter 1

Introduction

The advent of modern computers has set a new pace of development which is unmatched in human history. This phase of exponential growth has been marked with unusually large amounts of data which has opened the doors to such technology which could never be thought of a few decades ago. Everyone wants to get their hands wet in this flow, which has created a wave of digitisation of all sorts of information, from taking a simple note to highly sophisticated software and computations. Civil Engineering, on the other hand has been around since the dawn of human civilisation, with the modern discipline in existence since the early 19^{th} century. The architecture, engineering and construction industry has for a long time stuck with the conventional ways of doing things. But now it has gradually started to move with the flow. This process started with the digitisation of drawings in the form of CAD, and has now developed into a more sophisticated modeling paradigm, which is known as Building Information Modeling (BIM), where the aim is to create a virtual model of all the systems/features present in the building. It is noteworthy here that a building can mean any kind of civil engineering structure, although primarily residential, office or industrial buildings are referred to.

This kind of a BIM system aims to serve the building for its entire lifetime – from the very conception of the idea to the demolition or recycle, passing through various kinds of operations, maintenance or even renovation works. But, as has been observed from the literature, most of the focus is laid on the construction process. So, we will start with an example from the same. Suppose in the construction of a building, separate teams have been assigned the job of carpentry and masonry, which is generally the case. Now, let's say three separate carpenters C_1, C_2 and C_3 have been assigned the work of doors, windows and flooring respectively from within that team. From the masonry team, M_1, M_2 and M_3 have been assigned the work of footings, walls

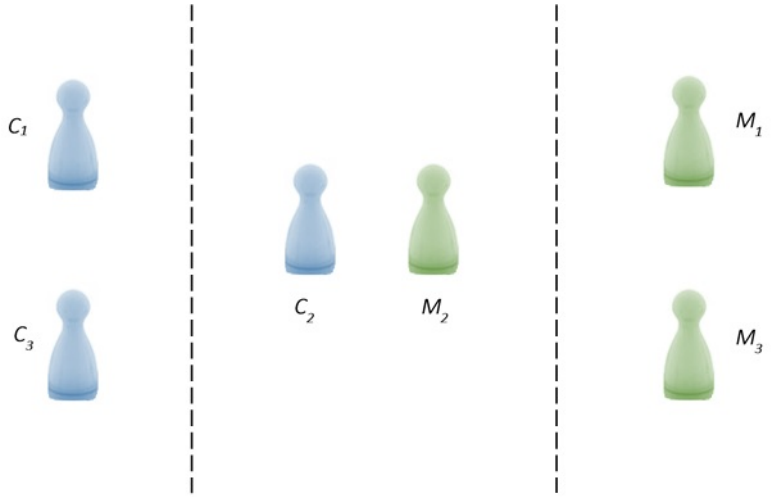


Figure 1.1: Understanding Granularity

and slabs respectively. Now, C_2 wants to coordinate with M_2 to fit a window frame within an opening, while the wall is being built. If for this process, all the six people are summoned and then C_1, C_3 and M_1, M_3 sit behind a curtain, and only C_2 and M_2 talk with each other, how illogical would it seem to have called the other four people (Figure 1.1). Now, just assume when the size and number of teams gets larger with more specialised work distribution within them, and everyone is called aimlessly for every discussion between members of different teams. How unproductive a situation it would be, with no one being able to properly do his work, and a lot of time being wasted even for petty issues of work overlap. A similar situation occurs on the level of inter-domain information, and all the domain models are loaded to sort out even small things between them, which can easily be sorted out by summoning (or loading) just a couple of elements. This is what granularity in building models is all about – providing only what is necessary and saving resources in this effort.

Now, this does not remain limited to the construction process only, but such a scenario would occur in every lifecycle stage of a building. This would involve people-to-people, people-to-machine, and machine-to-machine interactions, with everyone having different areas and levels of expertise. Thus, it is important that the concept of granularity be understood well and applied wherever need be in order to increase efficiency and ease the workflow.

1.1 Motivation

The concept of relevance and granularity of information has become really important in this age of information overload. The huge amount of information that is being produced every passing day [Marr, 2018], despite of its richness and numerous existing and potential benefits, also poses a great challenge as to how it should be handled and utilised efficiently. Information retrieval is an important aspect of this utilisation process, as we can experience in our day-to-day lives; from searching a simple thing on the web to addressing issues related to big data analysis, all require retrieval of data in some form or the other. In such a scenario, it is of great importance to identify the correct requirements of data and then provide solutions to fulfil those requirements in the most efficient way possible. This is where the concept of granularity kicks in. We not only need to get the right information but the right amount of information and in the right context, meaning that the information seeker has an option to go beyond what has been presented to him and dive deep into the topic based on what has been presented.

This abundance of information has effected almost all the disciplines of professional life. Thus, it is only natural that the Architecture, Engineering, and Construction (AEC) domain does not remain untouched by it. Building Information Modelling (BIM) has been the key element in this process for the AEC domain. As of the current scenario, BIM is a rapidly developing technology across the industry as well as the academic discourse. Its adoption has been constantly progressing on all scales, from implementation within organisations to development of national and global standards, legal protocols for liability issues, BIM certification, training and education [Smith, 2014]. This progress has been fuelled by the huge amount of opportunities it provides in terms of the technical superiority, better knowledge management, interoperability, building lifecycle analysis, economic benefits, and better decision support mechanisms among many others, as identified by Ghaffarianhoseini et al. [2017].

But what is missing in all this discussion is a fundamental definition of BIM itself. At first, it might seem that BIM needs no introduction, but there are different perceptions among the industry professionals as to what BIM actually is. There have been a number of efforts to define BIM, and it seems that most of them agree to the fact that it involves a digital representation of the building lifecycle process which is often rich in information. For instance, the BuildingSMART alliance describes it as “a digital representation of physical and functional characteristics of a building”, and “a shared knowledge resource for information about a facility”. This perception is slowly building

towards considering it as a more inclusive paradigm within the AEC industry, rather than just another tool.

Having stated the above, in the scope of this thesis, BIM would mainly be considered in context of a building information model, i.e., a 3D model on a digital platform, which is enriched with semantic information (see 2.2.1) about the building, along with the geometry. Such a model is the natural outcome of any BIM-based design process, and it serves as the central resource for the different professionals involved from the initial conception of the building to the end of its lifecycle. It is to serve this purpose that a lot of commercial software packages have surfaced in the market which provide a platform to host these resource rich models, and to carry out design and building process with ease. But these have limited the building information to software specific platforms, which hinders the process of standardisation and interoperability, which in-turn has led to more difficulty in information access. Despite of all their drawbacks, these software tools have still served some good purpose, more about which would be discussed in 2.3.2. Needless to say, a common standard of building information would provide a much smoother information access to all. Industry Foundation Classes (IFC) has helped realise this standardisation unto a great extent.

Another important point to mention here is that the BIM model actually consists of several partial/aspect models which are maintained with the perspective of different domains related to the building design and construction. Although BIM has been primarily incorporated within the construction phase of the building lifecycle, there have been continued efforts in recent years like [Becerik-Gerber et al., 2011], [Codinhoto et al., 2013], [Dong et al., 2014], [Motamedi et al., 2014], [Zibion, 2018] to explore the inclusion of the operations and management phase within this BIM paradigm, something which has been claimed since the very conception of BIM, but implemented less so. Thus, we see that semantic data like scheduling times, costs, energy data, data related to facilities management, etc., which is being generated all the time during the building lifecycle, gets added on to the bulk of information in the BIM models. All this has led to an over-increasing information within the BIM models, which has led to problems in handling and utilisation. With this, we again come back to our starting point about how granularity in information retrieval is important. Törmä [2013] has also identified problems related to multiple partial models in the process of BIM lifecycle, which are largely inter-related, but lead to a loosely coupled information pool due to lack of information exchange between models. He further argues that IFC (see 4.1.1) has been able to address this issue of interoperability only partially, and presents a case for linked building data. Linked building data is a kind of granular data, which is linked together, and can be accessed and

utilised granularly using unique identifiers. Our case is quite similar to this in principle but addresses another dimension of development.

1.2 Identifying the Research Gap

There has been an effort by the World Wide Web Consortium (W3C) towards the development of a semantic web which largely focusses on the interoperability and granularity of information in the domain of the web. This concept of semantic web goes hand in hand with linked data, as pointed out by Törmä [2014]. Figure 1.2 shows the evolution towards the web of data, and how it is similar to the evolution of the World Wide Web (WWW) as we know it today. It is as a subset of this broader sphere of linked data that we can think of a paradigm of linked building data. Undoubtedly, BIM has a key role to play as a foundation for this, especially the open BIM or the IFC. There have been continued efforts towards the realisation of this goal by organisations such as BuildingSmart, and also by independent researchers like Schevers and Drogemuller [2005] and Beetz et al. [2009], who have tried to migrate the IFC ontologies to those based on the Web Ontology Language (OWL), which is the standard for semantic web. Thus, we see that there has been an effort towards granularity in building data from the information infrastructure developers, which has led to a departure from the IFC towards the graph based data structure of Resource Description Framework (RDF), with the development of the IfcOWL schema. One of the reasons for this is also to get rid of the large monolithic package which the IFC file offers and transfer it into a more granular, semantic form. But even this transfer to RDF has shortcomings of its own, which has led to further development of simplified schema, which will be talked about more in 2.2.

What we have discussed until now is about the modularisation of the information in BIM models and its subsequent interoperability. This information can be utilised to serve all sorts of purposes ranging from designing, scheduling, budgeting to facilities management. But an important component which has been missing from this discussion so far is the visualisation of building data through this information. This also forms an important component of the information retrieval process, which is the representation of retrieved information, and the same will be the primary focus of our work. A significant part of the information contained in the BIM models relates to the geometric information of the building. There is no argument about the fact that visualisation of building data holds primary importance, and its significance has been talked about since the early days of BIM, as can be seen from the literature itself. A report from CRC Construction Innovation

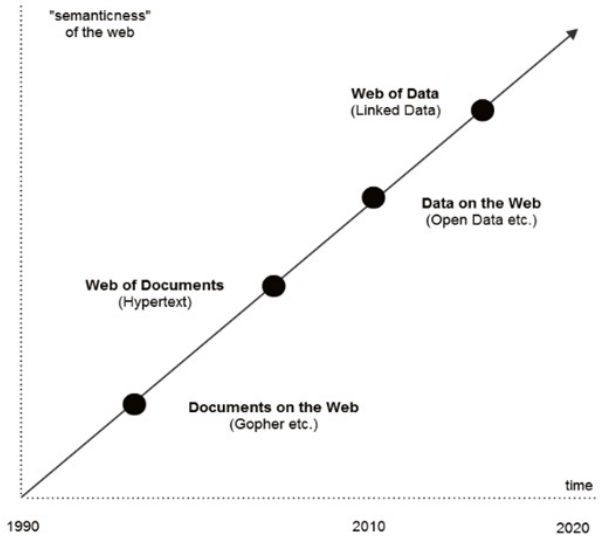


Figure 1.2: Towards Granular Web [Bauer and Kaltenböck, 2011]

[2007] states that "The key benefit of a building information model is its accurate geometrical representation of the parts of a building in an integrated data environment". Azhar [2011] in his review of trends and benefits of BIM, mentions it as the first application of a building information model.

While we see that the effort for granularity of data and its interoperability has been extensively researched and developed, there have been only limited efforts in the direction of implementing this granularity in visualisation of this data. Commercial software packages are mostly used as of now for the purpose of visualisation. One of the problems with using these is that they are expensive. Further, the files in IFC open standard need to be converted to software specific formats for visualisation. Thirdly, granularity is missing from the process of visualising. To make up for granularity, many of these software try to work around the problem by providing features for filtering different objects according to requirement. Although this strategy works well as far as viewing is concerned, it misses the whole point of incorporating granular approach within the process of visualisation, as we still need to load the entire model initially. There have been efforts for developing open source viewers for BIM data from IFC files, and many other independent efforts to address some of these problems, and these will we talked about further in section 2.3.

Despite all these efforts, what we see is that there has been a widening gap between the increasing granularity of BIM models and a lack of its visu-

alisation. Furthermore, the efforts for developing simplified schemas have led to the omission of geometric data altogether from the building data graphs. Thus, we feel that there is a need for this gap to be filled. We try to provide a means for the same by introducing a granular approach for visualisation of BIM models.

1.3 Scope of the Thesis

One might argue at this point that why is granularity even important in visualisation. There are a range of reasons for that, and we will try to present some of them over here. Any building consists of multiple domains, like architecture, structural, mechanical, engineering and plumbing (MEP), heating, ventilation and air conditioning (HVAC), etc. These are represented by separate IFC files in general (partial models talked about earlier). So, cross-domain interaction in visualisation is limited to loading the entire models first and then proceeding further. We cannot extract individual elements from different models and view them independently, unless we provide granular access. There are several other situations where the benefits of such an approach are clearly visible, and these have been talked about further in chapter 6.

Recent work by Zibion [2018] has been done in this direction for the application in facilities management, in collaboration with VisaLynk-Oy, Finland. But that work is primarily limited to 2D floor plans, leaving the benefits of the 3D model. We aim to provide through this work a contribution towards filling this gap in the granular visualisation of 3D BIM models, which would aim to complement the work done by Zibion [2018] by providing the third dimension on top of the 2D visualisation, but not limiting to the scope of facilities management. Rather, we tend to provide a more generic approach towards granular 3D visualisation, and try to fill the gap between the fast developing granularity of building data and a lack of its visualisation.

Keeping in mind the prospective development of the semantic web in the coming years and the growing popularity within the research community of linked building data as a subset of it, we have proposed a solution to break down the IFC files into individual geometrical entities which can then be modularly utilised for the purpose of visualisation. This provides a novel approach in visualisation which has not been found in any previous works. Further, the semantic information contained within the IFC is preserved through the use of a graph database. The use of graph based data structure is really important to note here. First of all, the data inside IFC file is highly inter-related, and storing it in a graph database is a strong

concept. Moreover, the proposed representation of semantic web and linked building data rests on representing data in the form of RDF graphs based on the OWL Schema. Thus, the concept of visualisation in relation with a graph database would prove to be beneficial when looking forward to incorporate visualisation of BIM within semantic web. Although that is not the aim of current work, we aim to provide a step in that direction by incorporating graph data structure. Furthermore, most recent efforts for simplifying the BIM data like SimpleBIM [Pauwels and Roxin, 2016], Building Topology Ontology (BOT) [Rasmussen et al., 2017] have used a similar approach to semantic data representation by removing the geometric data from the graphs. By providing a similar method for semantic data representation using Neo4j graph database, our work is aligned parallel to the current mainstream efforts, and thus, has the potential to provide a general approach for further efforts in this direction.

Also, since the concept of linked building data is based on the web, the visualisation must be rendered on a web platform. For this purpose, we chose to use WebGL based renderer, which provides us with multiple benefits in terms of efficient rendering of 3D models through web browsers. The use of IFC as a starting point can be justified as it is the open standard for representing BIM models, and has had a long and robust development history. It would also be good for the purpose of interoperability among other benefits. A detailed discussion about the various technological tools employed in this work has been presented in chapter 4. Naturally, no human effort is perfect in itself. Same is the case for us, and this effort also has some limitations which have been highlighted in the discussion section.

1.4 Research Questions

Having identified the research gaps, and the scope in which this work is proposed, we lay out primarily two research questions which we try to answer. The implementation of a prototype solution will also be done as a part of exploring these questions.

1. How can granularity be incorporated within the visualisation process of 3D BIM models?

2. How does granular visualisation help in the AEC/FM industry?

1.5 Research Methodology

The purpose of this research is to investigate a new approach for 3D building visualisation within BIM paradigm. Design science research methodology was adopted for this work. We tried to develop a prototype as a proof of concept for a novel solution approach towards addressing the problem of lack of granular visualisation in 3D BIM models. We started by establishing the problem context and relevance, identifying the gaps in the current research and development efforts. The goal of current effort was hence identified. Then, a conceptual layout was prepared on how to achieve the set goal, and efforts were made towards realisation of a prototype to support this goal. Henceforth, we identified the requirements for the prototype implementation which led us to choosing the required technological tools for the purpose. These were then implemented on experimental levels to provide confidence in the proposed concept.

As far as evaluation of this method is concerned, the very nature of it to be a web-based visualisation tool puts certain restrictions on the memory availability for it. But that has not been the aim of this work anyways. We do not try to increase the memory availability for web based visualisation, rather we work the other way around and try to minimise the memory requirement itself, thus increasing memory efficiency. Further, comparing the performance of this method with others which load entire models would not be fair as we are moving away from the concept of loading entire models. Thus, we would stick to descriptive evaluation methods for this work which will be covered in the discussion section. Further analytical and observational evaluation has been left for future studies.

Naturally, design is not a one-way process and there is required an iterative process consisting of a feedback and re-design mechanism. We just claim to have taken a step to fulfil the first iteration of this larger process, and further iterations will be possible after extensive evaluation of performance and utilisation, which have been left as a part of future work. We have, however, tried to put forward an interesting discussion related to this concept and its possible benefits in the industry from the initiation of the building to the end of its lifecycle, which has in-turn been the true motive of the concept of BIM from its initial days.

1.6 Thesis Outline

Chapter 2 provides a background of the problem domain, and prepares a base for the proposed solution. It is divided into three parts, which explain three

different aspects leading towards the build-up of the problem. Discussion from the literature in the form of domain knowledge and the pros and cons of the current efforts have been taken up. Chapter 3 then lays out the envisioned concept for the solution, along with the challenges and requirements that are a part of it.

Further, chapter 4 gives out a detailed review and explanation of all the technological tools which were necessary for implementing the solution and the context in which they were applied. After this, the actual implementation process has been discussed, including how the various challenges were faced and the requirements fulfilled. Then in chapter 6, analysis is made in terms of the limitations of the work, how it contributes towards the related knowledge domains, and the possible applications and use cases in the field of building construction and lifecycle management. We conclude after that providing a summary of the current work, and a direction for future work.

Chapter 2

Background

This chapter talks about the related knowledge domains on which our work is built, and provides a base for understanding the concepts which are explained later. The construction industry often faces issues related to interoperability of information. The information, on the other hand, is getting more and more diverse and complex with domains like energy simulations and carbon emissions, etc. adding additional load to the already bulky building information. This leads to the importance of concepts of data science like information retrieval and relevance to be applied in the context of construction industry. Also, many of the applications related to building lifecycle management do not require all of this information. Rather, granularity is sought after for the tasks which are targeted at specific locations or parts of the building for both construction and post construction operations. The workers on the field, for example, don't need to know what's going on at some other part of the field, unless it's directly related to their own work. The same would hold true at some level for the supervisor, or the sub-contractor as well, whose work is limited to only a particular section, or time of the construction.

Finally, all this information needs to be transmitted to various personnel and stakeholders involved in the process, and it needs to be in a form which can be easily interpreted by the recipient. 3D visuals are a great way for easy understanding of information. Especially when it comes to buildings, not everyone is trained in reading engineering drawings or 2D plans, and having a more realistic representation is better. So, having a granular approach to 3D visualisation seems relevant and that is what has been built up in this chapter.

2.1 Information Retrieval (IR)

Information is the second term within BIM, and when it combines with term building, we get a huge expanse of information which relates to the building lifecycle. Modern buildings, including all forms of built facilities like roads, dams, etc. are some of the most complex things which we create, and the teams involved in this creation process are even more complex [Crotty, 2013]. There is a large data source comprising of multiple documents of various types which are constantly being revised and updated by multiple people, and which require to be transferred between several teams, and firms who have different backgrounds and requirements. Then there is involvement of different individuals during different time periods, which further complicates the matters. Thus, we can see how we are getting overloaded with information in this digital age, and how buildings are a huge source of such information. It is in such a scenario that Information Retrieval (IR) becomes critical and trying to understand and improve it becomes important. Let us start with defining it first, then discussing about its characteristics and how they are related to our case.

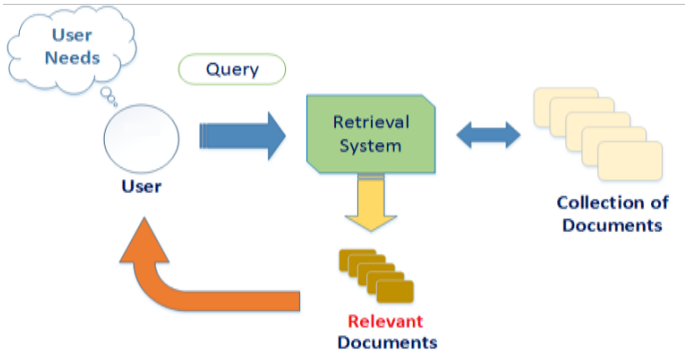


Figure 2.1: Information Retrieval Process [Information Retrieval Lab - CSUI, 2016]

2.1.1 Definition and History of IR

The term 'Information Retrieval' can be used in a very wide sense. Any situation which requires us to extract any kind of information from its source can be broadly termed as IR. For example, if we ask the name of a person, it might also be considered as IR as we are extracting the information of his name from his memory. If we look into the literature, we find that IR has a long history in terms of an academic field of study, and that is the

IR we are primarily concerned with for the scope of this work. Manning et al. [2010] have defined it as *'finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)'*. It is clear from this and many other similar definitions that IR in the modern world primarily refers to computer-based IR, which is also fair because that is where the major source of all information lies.

The discussion of IR as a method to retrieve information using computers goes back to the mid-20th century, with people like Holmstrom [1948] developing some early descriptions of computer search models. By the 1970s, many models had been developed and tested, like the Cranfield tests, but these were still limited in terms of the amount of data which they served. It was only in 1990s that web search engines appeared which served large data. But one thing is visible from all these efforts that the idea of relevance of information was present from the start of this discussion on IR, and has been core to it.

2.1.2 Characteristics of Retrieved Information

When we look at IR, we see that it can be classified and evaluated based on a number of criteria. The first can be seen as the nature of the information itself, from which we want to search. It can be in the form of a structured data, like most of the relational databases, or it might be unstructured data, like many things found in the real world. In fact, no data is purely structured or the otherwise. Rather, the IR process faces a semi-structured data scheme, and identifies documents and patterns (structure) from within that based on the queries.

Another aspect is the scale of the retrieval, which could range from looking something up on a personal computer, an enterprise or institution level framework, to a web-based search. All three require various levels of security, accuracy, architecture, etc. Yet another important aspect is the kind of the access, which is more related to the domain of application. During the earlier days of IR, it was considered to be mostly focussed on jobs such as librarian, clerks, etc. who were required to be professional searchers. Even with the advancement of computing technologies in other fields, this perception was only totally removed once and for all with the inception of the World Wide web in early 1990s, and with which came the web search engines [Baeza-Yates et al., 1999]. This expanded the scope of information retrieval beyond comprehension, and is still one of the major user of IR.

With the dawn of the digital age in the 21st century, most traditional disciplines went more or less digital in terms of their data storage and retrieval.

Same was the case for the AEC/FM industry. The retrieval of information from the resource-rich BIM models for exchange, interoperability, and communication between various stakeholders gained significance. We found that the following characteristics prove to be key when looking at the retrieved information.

2.1.2.1 Relevance

When we look at the history of IR, we see that relevance of the retrieved information has been the core purpose of the process. This is the reason that Cooper [1971] mentions that the concept of relevance lies at the heart of the problem of intellectual access. But what consists of relevant has been another challenge which has invited multiple efforts to define and develop various techniques time and again. Cuadra [1967] defined relevance as *'Relevance is the correspondence in context between an (information) requirement statement and an article, i.e. the extent to which the article covers material that is appropriate to the requirement statement'.*

Another thing is the criterion on which the relevance of an outcome is judged. A document might contain the queried keywords, but still not fulfill the purpose of the user. Should such an outcome be considered relevant or not, is another sub-question in itself, the answer to which is more inclined towards no. This has led to the development of a lot of techniques for sorting, ranking, filtering out the results to get relevant outcomes like [Salton et al., 1982], [Deerwester et al., 1989], [Page et al., 1999], etc. Thus, we can say that a relevant search outcome corresponds to giving the user what he requires, rather than just matching the keywords. This also goes hand in hand with the idea of granularity, as it entails serving only what is required and filtering out all the other results.

2.1.2.2 Speed

There is no doubt to the fact that time is one of the most valuable asset for human beings. So naturally, when information retrieval becomes more and more a part of our daily lives, the question for speed of the process is inevitable. Speed of search is essential to the modern day search engines. This is evident from the fact that references to the speed of retrieval (About 2,380,000 results (0.11 sec)) are shown along with the search results to instill confidence in the users. Many researches have been carried out by varying the speed of retrieval and observing the users like [Schurman and Brutlag, 2009]. These have found that there is a decrease in the number of searches per user with an increase in the search time. The site speed was even implemented as

a criterion for ranking of pages in a search result [Singhal and Cutts, 2010].

But there is another side to the discussion of speed, i.e., it adversely effects the quality of the search results. The search engines have to compromise on the relevance of the results in order to get them quickly. This idea has been put forward by Teevan et al. [2013] who have introduced a *'Slow Search'* mechanism, in order to achieve higher relevance. Thus, we see that these two ideas are of opposing nature, and we need to find a balance between them in order to get useful results.

2.1.2.3 Context

The idea of context is a part of providing relevant information and has been discussed since the 1990s. Belkin and Croft [1992] showed that the information need depends on the moment when the search is performed, and is thus contextualised by time. Similarly, there are can be many other factors like location, user information, history of search, etc. effecting the context of information. These may be considered as generic context. There can also be query specific context, which corresponds to providing information linked to the topic of search, while not missing on the relevance of the search itself. This can be understood in our case of BIM models as providing information of elements which are linked to a particular building element which is being queried primarily. Thus, providing semantics can be considered as a contextual information. This gives the user the opportunity to dive deep into a topic with only limited information being provided as a part of the search results, thus making them more comprehensive.

2.1.3 Granularity without Losing Context

Having talked about the characteristics of information for retrieval in general, we now discuss about these ideas from the perspective of our case and how they are related with the discussion of BIM and the construction industry. The idea of granularity, or limiting the results seems to go against that of comprehensiveness on the outside. But, in-fact actual relevance of results requires them to be comprehensive, without overflowing the user with additional information.

When we look at the AEC industry, and BIM models in particular, we find that they contain highly connected data. Also, people from many fields are accessing the same data, and their relevance varies according to their job requirements. So, they desire different aspects of the same data. Also, many important stakeholders across the lifetime of the infrastructure are not trained in engineering or design, and handling information within 3D or even

2D models might get difficult for them. In such a scenario, if we overload the users with irrelevant information, this might lead to inefficiency and difficulties in their working. For example, if we take the case of operations and management practitioners, Zibion [2018] mentions that these people are not used to handling 3D BIM models, and feel that it does not serve their core purpose. Rather, they prefer the conventional CMMS tools, which are primarily detached from the BIM paradigm. If we provide targeted partial models focussed on individual tasks, it would help integrate such works within the BIM framework reducing these problems to a large extent, and might also help to remove the apprehension from professionals from other fields to use centralised BIM models throughout the lifecycle of the building.

When we try to provide such partial models, another problem which arises is about the context. Isolated partial models can prove to be insufficient if the person concerned with them does not have enough knowledge about how these elements fit inside the entire system. The semantic information of the BIM models comes to rescue over here, providing the necessary context to the partial models, without the need to actually load the entire system. Thus, we find that the BIM models inherently have the potential to serve partial models from within themselves. There's just a need to exploit it. At the same time, there's also a need to strike a balance between the levels of granularity and the comprehensiveness which we expect.

We have left out the discussion about speed in this section, and it seems that reducing the size of the models might only lead to enhancing the speed with which they serve as the memory and hardware requirements become lighter. Even if significant benefits are not observed in the speeds, it would certainly not lead to a decrease in the speed. But this is only a secondary consideration in our work, as we are primarily concentrated on providing the feature of granularity.

2.2 Granular Information in BIM Context

Now, we take the concept of granularity and dive into the AEC industry, with the aim to explore how this concept has been taken up within this community by the different professionals. We also look into some of the state-of-the-art mechanisms currently available for incorporating granularity of information within the BIM paradigm, while discussing how all of it actually evolved. We will also touch upon the different types of information available within the BIM models, and how they are connected to each other, and to our problem and the proposed solution.

2.2.1 Granular BIM - Linked Building Data

2.2.1.1 Towards Granular Web

We had discussed briefly about the Semantic Web, and how we are moving towards it as shown in figure 1.2. Here we would like to expand upon that idea and see how it relates to BIM. The semantic web is an effort towards developing a web of data where all kinds of data would be able to be analysed using computers [Berners-Lee and Fischetti, 1999]. Both the definition and the technology have developed over time and the W3C [2011] mentions it as a common platform for sharing and reuse of data across domains and applications.

The current web is mostly about the interchange of documents. In contrast to this, the semantic web proposes the idea of connections between data, where data from different sources can be brought together in a common format under a common platform, and where individual applications do not keep the data to themselves. Rather, the data from a banking platform, for example, can be clubbed with that of a calendar application. A

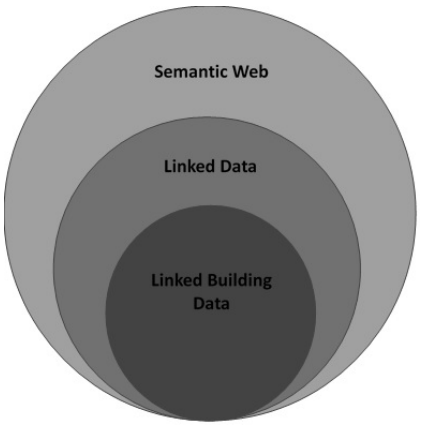


Figure 2.2: Linked Building Data as a Subset of Linked Data

very similar idea is also present within the BIM paradigm. Although limited mainly to the AEC/FM domain, BIM still works to bring every related thing under a single umbrella. For example, the scheduling and budgeting, civil and architectural designing, operations management, were all originally done by different specialised applications which did not dynamically communicate with each other. Within the BIM paradigm, they are connected to each other in real time through a common pool of information. Thus, it seems that the idea of BIM is in-line with the semantic web. The remaining disjoints are further filled by the idea of linked building data (LBD), which can be thought

as a subset of the larger pool of linked data (Figure 2.2), which in-turn forms our semantic web. The discussion about these data structures follows next.

2.2.1.2 Semantic Information within BIM

Before moving towards the data structure of the semantic web and that of linked building data, we think it is important to touch over the topic of semantic information and how it is represented within the BIM paradigm. The construction industry is information intensive, and BIM provides a medium for the exchange of this information [Zhang and Issa, 2011]. But this information is present in the form of several partial or aspect models, which represent the same facility from different perspectives. These are very loosely coupled pools of inherently related information, and efforts to semantically connect these using the concepts of linked data have been made by the likes of Törmä [2013]. We find that IFC has been able to achieve this goal to a large extent, by providing schema definitions for over 600 entity types. These entities are linked to each other through standalone relationship entities which detail the connections. An example of such a connection is shown in figure 2.3. More about the IFC schema has been discussed later in 4.1.1. But even the IFC format has not been able to fully serve the purpose of representing semantic interconnectedness of the building data, because of its too generic format, which makes the process of information retrieval difficult. Thus, there has been a shift away from the IFC schema towards lighter and easily parsable formats, while still keeping it as the base.

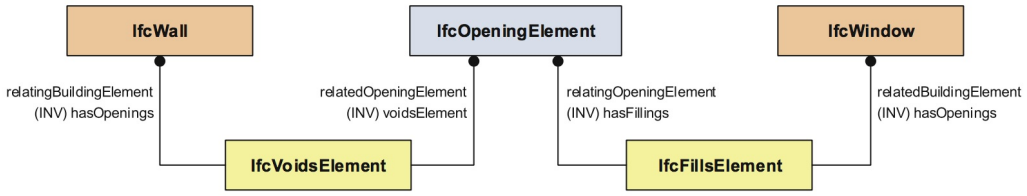


Figure 2.3: Example Representation of Relation within IFC [Borrmann et al., 2015]

2.2.1.3 Graph Based Data Structures

Most of the newer formats which are emerging are based on representing data within graph-based structures. Graph databases are a type of NoSQL databases and have recently gained popularity, especially in situations where the data is highly inter-connected, and thus storing and retrieving it through

relational databases becomes more and more expensive. Same is also the case with the semantic web, where Resource Description Framework (RDF) graphs are being used to represent data and its connections in a common format. The RDF is represented in terms of triples comprising of two entities and a relationship connecting them. The conceptual schema for these is defined using the Web Ontology Language (OWL). The other side of the coin is linked data. Linked data refers to interlinking of data in such a manner that it becomes more useful for semantic queries, and the information is shared in a computer interpretable way. A subset to this is liked open data, which is available to reuse freely. Both semantic web (consisting of RDF and OWL) and the linked data are complimentary to each other with the former conceptualising the meaning of entities and the latter contextualising them [Törmä, 2014].

When we think of putting BIM inside this realm, a whole lot of opportunities of linking building data with linked open data from other fields open up. For example, application areas like land registry, road infrastructure, etc. will develop a completely new understanding. One of the major benefits of graphs is that it has made information granular. All the entities are in the form of nodes in a graph which can be accessed individually without bothering other entities. Other benefits of graphs include easy interpretability and allowance for far more complex queries than the regular relational sets. Thus, the way of the graph is being taken up in the AEC industry as well, which has led to the development of a number of data models as discussed below.

2.2.1.4 Evolution of Granular BIM

BIM has been around for a while now. The efforts to tackle problems of interoperability in building data have been made time and again, and are still progressing. IFC was a great step in this direction. Since then the ways of data representation and the levels of their granularity have evolved. We were able to identify four distinct stages of such an evolution based on the literature in this domain (Table 2.1). The first one was obviously IFC standard, which is represented as a STEP file based on the EXPRESS schema of data definition. Since the EXPRESS language is not based on formal semantics [Krima et al., 2009], there have been efforts to translate it to IFC-based OWL ontologies. One such effort was the development of OntoSTEP by Krima et al. [2009] and Barbau et al. [2012], which was a more generic effort not focussed only on IFC, but still taking it as a reference. Because of the complex representation within the IFC, information retrieval has always been a challenge which has traditionally been handled by the specialised

software packages. The concept of linked data offered a solution to such problems which led to the departure from IFC towards LBD like Hoang and Törmä [2015].

The problems associated with bulky and proprietary software paved the way for efforts like Zhang and Issa [2011] who built simpler and direct retrieval mechanism from the IFC files using IFC-based OWL ontologies. These were not the first of their kind. The idea for converting IFC to OWL had been lingering around since Schevers and Drogemuller [2005] introduced it. It was further cemented with the development of IfcOWL in 2008-09 by Beetz et al. [2009]. The IfcOWL ontology quickly took the center stage in the LBD community by opening the IFC data to third party applications using the RDF graph representation. A number of IfcOWL structures were hence developed and there was felt a need to standardise and formalise the conversion process from EXPRESS to OWL for the construction industry which was done by Pauwels and Terkaj [2016]. The IfcOWL is semantically very similar to the IFC, and thus carried the baggage of the IFC in terms of complexity and bulky logics. Pauwels et al. [2017] identified two main directions for enhancing the IfcOWL ontology – splitting the ontology into diverse and easy-to-use subset modules, and serialising geometric data into less complex semantic structures. One of the efforts along this line was SimpleBIM by Pauwels and Roxin [2016], which made a number of simplifications including removing the geometric representation data. Mendes et al. [2015] also proposed a similar simplified ontology by the name of IFC Web of Data (IfcWoD).

These efforts for modified IfcOWL were limited to specific use cases, and there was a need for something which is more generic along with being light. This led to Rasmussen et al. [2017] proposing a simple Building Topology Ontology (BOT), only covering the core concepts of a building and providing room for extension to domain specific ontologies. Thus, we now have a generic and granular ontology which is derived from IFC, but does not carry its baggage.

2.2.2 Geometric Information in BIM

There is no denying that the geometric data in BIM holds utmost importance. This fact has been noted by many researchers throughout the history of BIM. Zhang [2018] notes that geometry data is not only important for the required visualisation in BIM tools, but also for many BIM researches that rely on geometric information. The aim of our work is also closely related to the geometric data of BIM. We intend to modularise the process of visualisation of BIM, just as the information is being granularised through the

	1 IFC file	2 RDF file	3 ifcWOD, SimpleBIM	4 LBD (generic modularity)
Granularity	Low	Medium	High	High
Complexity	High	High	Low	Low
Linking (semantics)	-	RDF graph	RDF graph	Modular RDF graphs
Direct Query	BIMQL	SPARQL	SPARQL	SPARQL
Serialisation Format (default)	STEP	JSON-LD / XML / TTL	JSON-LD / XML/ TTL	JSON-LD / XML/ TTL
Identifier	GUID	URI	URI	URI
Schema	EXPRESS	ifcOWL	Modified ifcOWL	BOT, PRODUCT, PROPS

Table 2.1: Evolution of Granular BIM

use of RDF graphs and OWL ontologies as discussed above. Even within the LBD community, there have been efforts like Pauwels et al. [2011] to represent geometry separately from the semantics and then link them over the semantic web. Thus, it is important to know that how this geometry is actually represented inside the BIM models, specifically the IFC and the RDF formats, and how it is developing further.

2.2.2.1 Representation within IFC

A major portion of the data within an IFC model is comprised of the geometry. This is essential in defining the physical entities of the building, performing structural analyses and visualising, etc. Inside the IFC, geometry can be defined using a lot of different constructs, based on the requirement. For simple visualisation, it can be represented as triangles and meshes. For more complex calculations, it may have constructive solid geometry (CSG) or NURBS representation. It even allows for element-specific definitions for certain building elements. For placement purpose, each element has its own local coordinate system defined by *IfcLocalPlacement*, which is defined relative to the parent element. This hierarchically relative system allows us to navigate along the structure, as shown in figure 2.3. Secondly, relative placement can also be represented using topological relations which are defined in the form of standalone relation objects like *IfcRelFillsElement* or *IfcRelVoidsElement*, etc. Elements can also be represented as extruded solid geometries, like the column shown in figure 2.4 which is extruded from an I-shape profile.

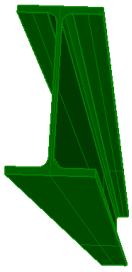


Figure 2.4: Column represented using extruded solid geometry [BuildingSMART-Tech, 2019b]

It even allows for the representation of structural elements, their material and joint properties, specifying loading patterns, etc. as shown in figure 2.5. The structural element can be connected to the point of connection using

IfcRelConnectsStructuralElements and to the loading activity using *IfcRelConnectsStructuralActivity*.

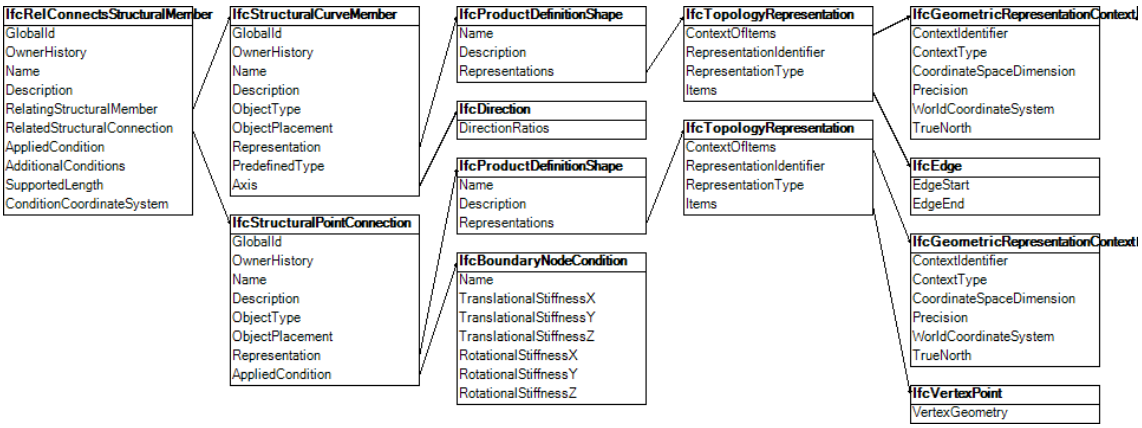


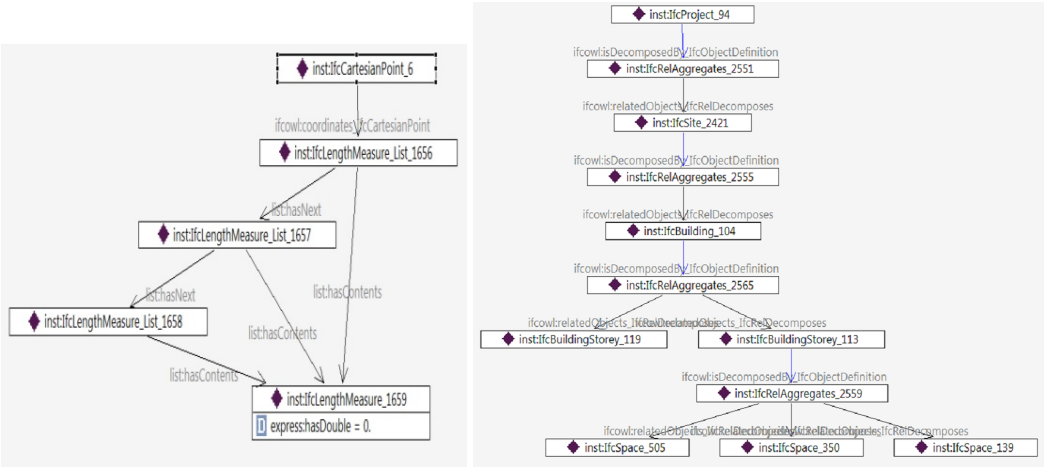
Figure 2.5: Structural member related to a point connection using *IfcRelConnectsStructuralElements* [BuildingSMART-Tech, 2019a]

2.2.2.2 Representation within RDF

The IFC provides extensive definitions of all kinds of geometry. But again the reasons for moving towards RDF have been mentioned previously. The representation within RDF (specifically IfcOWL) is semantically very similar to IFC. The IFC objects translate into OWL classes and are represented as nodes inside the RDF graph. The relationship objects of IFC also translate into nodes, and the relationships are represented as connections withn the RDF graph. Figure 2.6a shows the representation of a cartesian point within RDF. It is a translation from the concept of LIST within IFC. Figure 2.6b shows how there are muliple instances of the same *IfcRelAggregates* inside the RDF graph. Efforts like SimpleBIM [Pauwels and Roxin, 2016] and IfcWod [Mendes et al., 2015] tried to remove such inefficiencies and make the graphs lighter.

2.2.2.3 Later Developments and Current Work

We know see that newer models which are more granular and generic in nature are being introduced in the LBD community. The representation is still based on the RDF graph structure, bu these are being lightened and simplified for use. Many of such efforts have removed the geometric data altogether from the RDF graphs citing the reason that it is not used very often in linked data applications. One of the major highlights among these is



(a) Representation of a cartesian point

(b) Spatial Topology Structure within IfcOWL

Figure 2.6: Geometric Representation inside IfcOWL (RDF graphs) [Pauwels and Roxin, 2016]

the BOT onltology [Rasmussen et al., 2017], but it also has followed the path of getting rid of the geometry. Instead, there have been efforts to represent geometry separately on the semantic web like Pauwels et al. [2011]. These can then be connected to the other information over the semantic web itself.

The system which we use is similar in approach to the mainstream efforts. We translate the semantic information from IFC to a graph based data structure, Neo4j leaving behind the geometry which is later connected externally through a visualisation platform. Neo4j (see 4.4.1) offers a native graph representation of labelled property graph, and connecting it with a visualisation framework would provide a good start to connecting the building semantics to the geometry. This approach can further be extended to other types of graph databases like RDF, and can be of great help once the semantic web and linked building data becomes the norm in the industry and elsewhere.

2.2.3 Connecting Geometry to Semantics

We talked about semantics and geometry in BIM models, which are like the two sides of a coin, and compliment each other in all kinds of BIM related applications. We have already seen in table 2.1 how the granularity in semantics has evolved to make the information easily accessible to various stakeholders and applications alike. But the geometry has been sidelined in all this, and most of the visualisation tools are still using the conventional

ways, as we will see in the next section. Thus, even when we access the building information granularly, in order to visualise we still need to load the entire models. This gap becomes more visible when we try to work on inter-domain queries, for example. Suppose we want to work on a small part of an air conditioning system which is connected to certain elements of both the electrical and the HVAC models. In order to visualise such a thing which is small as compared to the entire building model, we need to load the entire models of all the related domains and then apply our query to show only a part of it. The semantic information like material properties, individual connections' properties, etc. on the other hand, can be queried granularly through the graph structure of RDF, for example. This situation creates a mismatch on the level of granular service of geometry and the semantics. We aim to fix this by proposing a granular way of visualisation, which will only load the required elements from different models, without requiring the whole of them. Further, it would be connected to the semantic structure to create a mapping between the two at the same level of granularity.

2.3 Visualisation of BIM & Current Efforts for Granularity

2.3.1 Importance of Visualisation

Humans are visual beings. Olivares [2013] cites the conclusion of a research which says that we process visuals 60,000 times faster than text. Figure 2.7 explains his findings better. This can also be verified by the fact that most of the content on the web today is visual in nature. Even the famous VARK learning model by Fleming [2001] gave primary prefernce to visual mode of learning. Thus, visualising 3D building models holds its own importance.

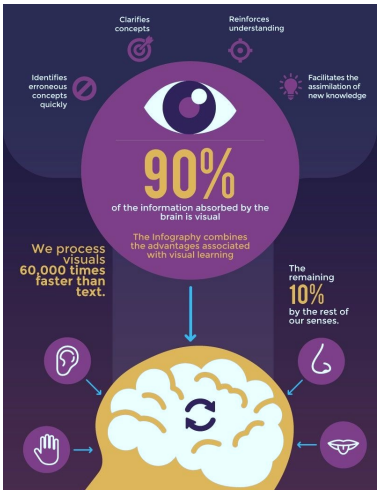


Figure 2.7: Humans are 90% Visual Beings [Olivares, 2013]

Even before the days of digital 3D modeling, architects used to create scaled physical models of the project in order to better convey their design intentions to the stakeholders. The digitalisation of these has opened a whole new set of opportunities, many of which are being realised through BIM. If the

visualisation was left out to the imagination of the people involved, everyone would have a different mental model of the project, which might cause problems in communication and execution. Additionally, 3D visualisation also provides other benefits like sellability of the project, improving speed and accuracy of the work, as the contractor does not need to spend time understanding the 2D drawings, and checking for possible mismatches. It might also be useful with the regulatory authorities who can have a better idea of the impacts of the buildings beforehand, leading to both expedited and improved regulatory process. These are the reasons that visualisation has been emphasised within the BIM paradigm from the early days like [Azhar et al., 2008], [Dawood and Iqbal, 2010]. The only drawback of 3D visualisation is seen in the increasing size of the models, which then requires better hardware and software to operate smoothly. Our aim would also be to tackle this problem, by reducing the size of the models while they are being used.

2.3.2 Current BIM Software - Are they really granular?

The BIM software solutions in the market can be categorised into commercial and open source solutions. Considering first the proprietary software, which often also provide cloud services to speed things up and reduce the load on client machines. This seems to be very good at first, but those viewers can only render their own formats. For example, Autodesk Forge Viewer can only render SVF format, similarly Bentley uses iModels. These are all closed formats without any open documentation. Although these software claim to support other formats like IFC, or even from other vendors, but they still convert it to their own format behind the hood in order to render. This limit on the format has actually bundled the cloud data storage facility to the model viewing platform. It is like saying that if we want to use Windows operating system, we can only use Internet Explorer as the browser. The cloud facilities outwardly appear to improve data interoperabilty, but on the contrary, they are further limiting it by introducing closed file formats. There is another level of problem which might occur between different stakeholders using different software packages. For instance, if a contractor uses Autodesk Forge which has its own viewer, and the subcontractor likes to use Trimble Mobile Viewer on the field, both of them would be stuck with data exchange problems even after having procured both the software solutions, thus raising the cost and time requirements for the project. Thus, it is necessary to have a common and open standard for easily viewing 3D models over the web which is free from these limitations.

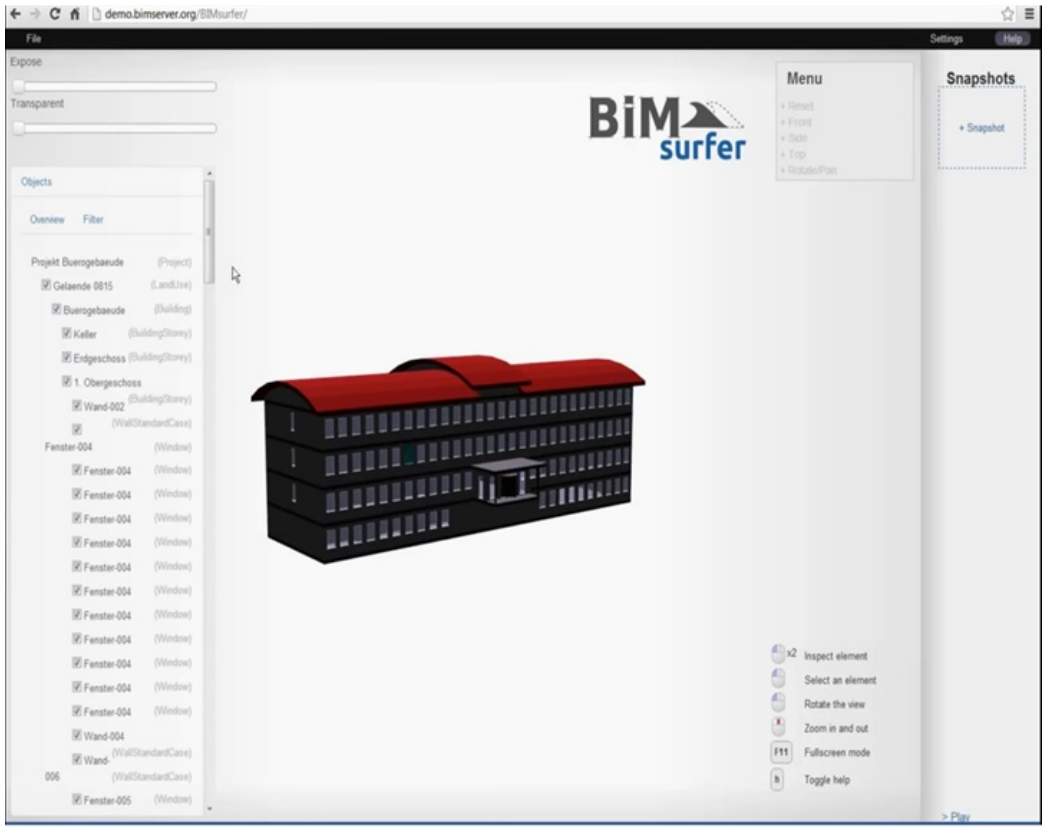


Figure 2.8: Example Interface of BIMSurfer Platform - Model loaded from BIMServer

Turning our attention towards the open source software, we have BIM Server as the main platform which has BIMviews as the default interface for viewing. Although it solves a lot of the problems of proprietary software by using open standard formats like IFC. But when we look from the point of view of granularity, even these don't fare very well. BIMSurfer, for instance is another viewer based on the BIM Server platform which loads everything with the first load as shown in figure 2.8. What many software like this do is serve a look alike of the granular functionality, without implementing it from the core. This is done by loading the entire model first, and then just providing a toggle to display or hide the required elements.

Mainly two kinds of loading APIs are used by the majority of software – Bulk Loading or Streaming. Most of the proprietary software use bulk loading, which loads the entire model in one go. It is then processed and converted to suit the format of the viewing platform, which is then presented to the user. As can be understood from the nature of it, this has higher

memory requirements apart from the problems of closed format mentioned above. Then we have BIMviews which uses the streaming method, wherein the data is constantly being transferred from the server to the viewer in real time, but still it's the data of the entire model.

Apart from these, there have been some independent research efforts towards the direction of granular building visualisation. A significant work for light-weighting the building models for web-based rendering was done by Liu et al. [2016]. Although they still worked on entire models as whole, without trying to use them partially. But they used a granular approach to remove redundancy from within the IFC elements. Their method provided good results for using large models with ease over the web despite the many limitations (like memory, etc.) that come with the web. Another effort which focussed more on granular retrieval and rendering of building models was developed for 2D floor plans by Zibion [2018]. It is one of the methods which actually incorporates granularity from its core, but it limits this to primarily 2D floor plans. Although the argument to use 2D floor plans for navigation like purposes is strong, but we still can not ignore the 3D altogether.

2.3.3 Possible Improvements for Granularity

We feel that there has been a lack of efforts in general towards the concept of granular visualisation in BIM. Many methods for handling geometry, and the web-based visualisation of BIM models have been suggested lately, and some of those concepts can be included for the making of a granular framework. What we suggest is to use partial loading, wherein the problems associated with large size are not present, and real-time updates can be done by reloading the partial models with utmost ease. Pauwels et al. [2011] suggested how 3D building information can be handled better over the semantic web, and in connection to the other related information. Hestman [2015] demonstrated the use of WebGL technology to display building models over the web. Pauwels et al. [2010] also connected the semantic information to the geometry by visualising it within a game engine environment, but it still wasn't about partial models. We take some lessons from each of these and apply in our case as explained in the next chapter.

Chapter 3

Proposed Solution Approach

This chapter aims to provide a detailed explanation of the concept employed as a part of this thesis. We have seen until now how the current solutions for 3D rendering of building models have adopted only limited approaches. We also identified the fact that granularity has been missing fundamentally from most of these efforts. Herein, we have tried to lay out some of the requirements for a granular visualisation solution, then identified the primary challenges in realising that goal, and have then gone on to discuss the envisioned concept, which is focussed around exploiting the linking between data, its effective retrieval, and a user friendly and easily to access interface for its visualisation. As this work, although primarily focussed on 3D models, also aims to incorporate an already built 2D solution, we think it would be good to also point out some differences in the implementation as compared to the 2D approach.

3.1 Requirements

For the development of an effective BIM visualisation solution, we ought to keep in mind the requirements from the perspective of different stakeholders and professionals who will ultimately be utilising that solution. Further, any acceptable solution should also keep up with the demands and trends coming up in the AEC industry. Working along similar lines, Morgan had identified five categories – hardware, software, user, researcher, and developer requirements for a visualisation software. Although his work was more generic in that he considered information visualisation as a whole, we can still get some useful inputs for our case of BIM visualisation.

It is important to note at this point that all the requirements do not have the same preference, and that some might be dependent on others, like our

hardware and software requirements will depend on how we want our user to access the platform, or how big the data size we want to render. This size of data would further depend on the client demands, etc. Thus, it would be a highly inter-related pattern, and it is crucial to identify the basic variables from within this. One such variable, for example, would be the resource effectiveness. It is in direct accordance with the principle of granularity, and focussed data retrieval. This is, in fact, a part of any design optimisation process. Apart from data retrieval, the method of data storage and querying itself can be of great significance. The building data is highly connected as we have seen in section 2.2. Thus the storage requirements can be laid down keeping that in mind, and moreover, how we can exploit those connections within data for the best results.

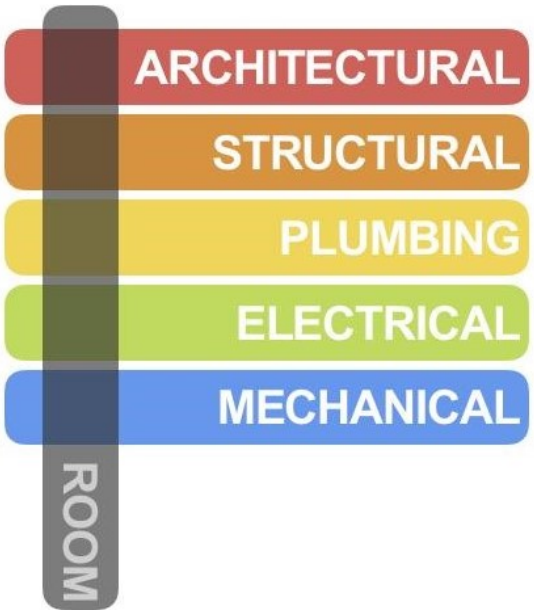


Figure 3.1: A room swipes across all disciplines [Zibion, 2018]

Another view of looking at this is through the lens of the use cases for which it is meant to be used. One of these is Facility Management (FM). Generally, the FM personnel require to work in operations phase of a building, and thus deal with the amalgamation of various disciplines like architecture, structure, MEP, HVAC, etc. which are usually independently developed during the construction phase. Thus, we see a room which requires some maintenance and which has elements from all different systems connected to it. In such a case, providing interoperability between disciplines in a granular fashion would be essential as is displayed in figure 3.1. On the other hand, if we look at construction progress monitoring, bridging this split might not

be as essential. Thus, we need to provide flexibility of including information from one or more IFC model files and combine partial data from each of them on a single platform.

One other consideration is the accessibility of the platform. If we want to feed the partial models to construction workers as a part of situation management, it should be easily accessible through their mobile devices. This leads us to look for more general platforms for hosting our solution than developing very specific software packages. Another natural requirement is then the smoothness and ease with which it can be used. The intuitiveness and ease of use should also consider the nature of the model itself. Despite all the recent efforts, 3D models, in general, seem to be more difficult to use for navigation purposes. More comparison between 3D and 2D models is coming in the following sections. For now, we need to understand that the user interface should be easy and intuitive for it to be widely accepted for use.

Obviously, one other major job would be to provide interoperability among different people. This should be taken care by the accessible nature of the platform, which is proposed to be web-based, and can be used with mobile as well as stationary devices. Another thing which is necessary for all this is vendor neutrality of the solution.

Thus, we can summarise our requirements for an effective BIM visualisation platform as follows:

1. Resource effectiveness in storage and retrieval of building data
2. Granular and inter-disciplinary access method
3. Accessible platform (possibly web-based)
4. Vendor Neutrality
5. Simple and easy-to-use interface

3.2 Primary Challenges

Having laid out the requirements and the concept for our visualisation platform, we now look at some of the basic challenges in its implementation. We have tried to only list down the challenges over here, and each one of these is then separately addressed in the implementation section.

The building models we use for our purpose are in IFC format (see 4.1.1). The IFC format stores geometry as an implicit information within it. This

implicit geometry needs to be converted to explicit form so that it can be visualised on any software platform. Thus, we need to extract this information using some pre-processing tools which will be discussed later.

The 3D models which we intend to use are generally bulky and complex. Such large sized models need to be simplified for storage and effective data management. We try to separate the geometry from semantics and then link them back after visualisation. This way, we ensure that the visualisation process can be optimised and the semantic information is also not lost.

Another challenge is to render 3D elements granularly. Once we have simplified the models and extracted explicit geometry, we need to render only what is required. We need to find a suitable technology for that, while also keeping in mind the relative positions of different elements which might get distorted when loading individually and independently.

After we have rendered the required, we need to provide a mechanism for selecting elements on click in a 3D environment, such that we can use their identity to retrieve further information about them from our semantics database. The creation of a semantic database is a separate problem in itself, but we will be utilising an already available solution for the current purpose.

Then finally, we would also like to integrate this 3D rendering platform with a 2D one, so that we don't miss out on any ease provided by 2D, as argued in the coming sections, while also maintaining the 3D with its own benefits. This problem can be approached in two ways, putting 3D inside 2D, or 2D inside 3D platform. More about this will be discussed in section 5.4.

3.3 Concept

This section introduces the envisioned solution approach which is being put forward through this thesis, and is developed keeping in mind the requirements put forward in the previous section. When we look at 3D building models, which are developed as a part of a multi-disciplinary collaborative process, we usually have an idea of models split by the domain. For example, we have different models each for architecture, plumbing, air conditioning, etc. But despite of this division, the models are still very large and are stored as separate IFC files for each discipline.

In order to reduce the size, provide interoperability and thus, manage them effectively and granularly, we propose to divide these models separating the geometric and semantic data held within the IFC files. The semantic data is proposed to be stored in a graph based data structure. This graph database offers a connected structure which facilitates the blending in of

the semantic building data, which is also highly connected in nature. This further helps in efficient retrieval of information which is difficult to query directly from the IFC files. The geometry, on the other hand, which is the vital component for our purpose of visualisation, is proposed to be broken down further into separate files for each geometric element (Figure 3.2). The

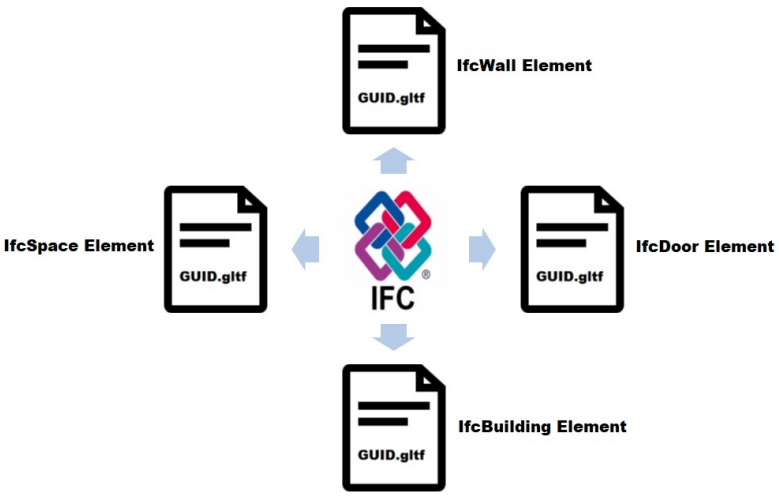


Figure 3.2: The IFC file is broken down into several smaller files, one for each geometrical element having a GUID

connection between them, however, remains intact through the unique identifiers for objects within IFC, the GUIDs (see 4.1.1.3 for more information on GUID) which are transferred as metadata to both the geometric files as well as the graph database. The next part of this work focuses on developing a web-based platform as a proof of concept for the ideas presented here. This would provide an interface for the user to enter his requirements, according to which the information would then be retrieved from the graph database and rendered to the user. This platform is also meant to be interactive, wherein further information related to the rendered elements can be accessed. A brief analysis of the different steps followed as a part of this approach follows next.

3.3.1 Creating Geometry from IFC

As we had mentioned earlier, the geometric data occupies majority of the space within the IFC files. So it only seems natural to break it down into its components in order for a smoother and more efficient management and utilisation. Also, if we just try to keep it as a single file, we do not get access to individual elements while rendering which defeats the purpose of granularity.

Thus, the individual geometry files are generated beforehand and stored in a separate repository from where they can then be retrieved on demand. This generation is done in two steps. First, the IFC file is converted into multiple COLLADA (see 4.1.2) files which are then converted to glTF (see 4.1.3) files. We use the IfcConvert application of the IfcOpenShell library for the first step, and Collada2GLTF converter, which is an open source command line tool, for the second step. Trials were first carried out using COLLADA format, which has an XML schema representation and relatively smaller size than other alternatives. While testing our prototype with COLLADA files, we encountered that glTF format was better suited for our purpose, which led us to its final choice.

3.3.2 Exporting Semantics from IFC

Semantic data, although not the main focus of this work, is an important part of the IFC file. This data is generally accessed through specialised BIM software which parse the IFC file for its retrieval. But it is not easy to get it directly from IFC files to a web-based platform, which happens to be our case. Thus, we store it in an external database as a part of pre-processing the IFC file. The choice of graph database (Neo4j in our case) as the external storage mechanism has been made keeping in mind the highly connected structure of the IFC file, which is difficult to model using relational schema, but inherently fits the graph data model. There have been efforts in recent years by Khalili and Chua [2013], Tauscher et al. [2016], etc. for the development of graph-based schemas and information retrieval approaches for the IFC files. An effort for automatic conversion of IFC models to labelled property graphs, using Neo4j was carried out by Ismail et al. [2017]. Another effort for such a conversion had already been developed as a part of previous work on this project by Zibion [2018] and we have used the same for our case.

3.3.3 Loading Geometry onto Web-based Platform

As we have already seen in section 2.3, most of the current open source efforts like BIMSurfer, BIMviews, DDS IFC Viewer, or commercial software like Autodesk Revit, Tekla BIMSight, ArchiCAD, etc. either use the process of Bulk Loading or Streaming for the purpose of loading BIM models onto renderers. We argue that these established methods do not serve well for the purpose of granularity. So, we move away from these methods and try to load granularly according to the user's requirements. This also enables us to manipulate individual element files separately and load only the required files. We use the WebGL technology, with ThreeJS as the high-level API

for the rendering. This is then integrated inside ReactJS framework, which is a front-end JavaScript library, thus providing a better and faster way of organising our content on the web page. The web platform provides an interface for the user to enter his requirements, and subsequently the graph database is queried to know what elements are required, which can then be identified and loaded from the repository using their GUIDs.

3.3.4 Connecting Geometry to the Semantics

We see that a lot of effort has been made by the research community towards the simplification and granularity of the semantic data, as we have seen earlier with the linked building data, SimpleBIM [Pauwels and Roxin, 2016] approach, BOT ontology [Rasmussen et al., 2017], and the related schemas like IfcOWL, etc. Even the efforts like [Ismail et al., 2017] mentioned in the previous section have considered only semantic data of the IFC files while converting it to a graph data structure, but leaving out the actual geometry from the picture, mentioning it only as a part of future scope. Thus, we try to fill the gap by connecting the two while keeping them on the same ground of granularity. The rendered elements can be clicked to select and display further information related to them from the graph database, thus achieving an interactive rendering of the partial models.

3.4 How is it Different from 2D Approach?

As mentioned previously, 3D models tend to get difficult to handle when it comes to navigation purposes. Although immersive 3D experience has been developed well in other fields like gaming, movies, etc. But it still requires higher capacity on hardware ends, and is not very easily accessible on multiple platforms. Our discussion here is focussed on the AEC domain and aims at providing vendor neutral and easily accessible rendering platform while utilising minimum possible resources. A similar reason was cited by Zibion [2018] for moving away from 3D rendering and towards 2D. But we argue here that it doesn't solve all the problems. 2D floor plans might work well when thinking about navigation, but once you've reached the required place, you would like to have a much more realistic view of the thing, especially when the depths of various elements located close-by vary drastically. For example, a light bulb and its switch inside a room might not be very realistically represented with a 2D plan, and more so, when it comes to identifying the electrical line connecting the two. A 3D model would simplify the understanding of such situations very well. We thus argue in favour of an

amalgamation of the 2D with 3D.

When we think about the differences in implementation as compared to 2D, we see that the 2D floor plans can be easily represented using Scalable Vector Graphics (SVG) format, which provides a DOM type structure and access, and which also has support of dynamic loading inside ReactJS framework. Thus, we can have a single file for a floor plan and load only elements of it on-the-go. On the other hand, for 3D models, no such support was available. Thus, in order to load separate geometrical elements of a model individually, we needed to find a way to access them separately, which has been presented in this chapter. All the 3D elements are rendered onto a single HTML5 canvas element, which in-turn does not allow us to dive inside it like the way in a DOM. Thus, we need to rely on other methods like capturing through positional attributes inside the scene on the canvas. Another thing of importance is that when trying to combine the 2D and 3D rendering platforms, there is a mismatch between the natures of rendering of the two which then needs to be addressed. All in all, we try to incorporate all the dimensions of geometric visualisation and more, moving in the same direction as that of multi-dimensional BIM which encapsulates concepts up to 7D BIM, as mentioned by Saxon [2018] in his article titled 'Getting the dimensions of BIM into focus'.

Chapter 4

Technological Tools

After establishing the context of the problem and proposing a solution approach for it, we now move on to another important aspect of our development process, which is the technological foundation on which our prototype for granular visualisation is proposed to be built. Such a solution encompasses a wide variety of technology ranging from digital construction industry to web development tools to database management. It is the combination of all these which leads to a better and wholesome solution for the industry. This chapter discusses in detail the various tools which were used for our development purpose, providing their relation with the digital construction industry and the context in which they have been applied here. They have been listed in the order in which they come into use, starting with the file formats for representing building data, the pre-processing tools used on these files, the web development tools used for rendering, and the graph-based data structure for data handling.

4.1 Data Models & File Formats

To store the building information in digital form, certain data models have been devised. These include all kinds of proprietary and open source formats. Some of these are designed to handle all kinds of data from geometry to semantics to external related information like costs, scheduling, etc. Others might specialise in storing only certain aspects of the information, like 3D file formats for geometry. Further, many data formats are generic in the sense that they are not meant just for building related data, but might serve for other domains as well. We are going to discuss here about three such formats which are relevant to our work.

4.1.1 Industry Foundation Classes (IFC)

Industry Foundation Classes is the open international standard for storing, representing, and sharing building information. It has been described as the standardised, digital description of the built environment, including buildings and civil infrastructure (buildingSMART, 2019). The IFC can store all kinds of data including geometry, semantics and topology. For example, it can codify the identity, characteristics and relationships like material, cost, thermal properties, connections, locations, etc. of the objects like columns, slabs, beams, etc. and the processes, people and organisations connected to them. The IFC was conceptualised in the late 90's with the main focus on interoperability, exchange, collaboration, vendor neutrality, and standardisation of the digital building data in the Architecture, Engineering and Construction (AEC) industry. Before the adoption of IFC, the various stakeholders like the architect, structural engineer, owner, contractor, who were generally using vendor specific software, had a hard time communicating their ideas with each other. For example, an architectural model developed using ArchiCAD could not be opened by a contractor who was using Autodesk Revit. Thus, the IFC provided a middle-ground for the stakeholders to effectively exchange their information. As of present, over 150 software applications support the exchange of IFC data according to the buildingSMART International, which is the organisation responsible for developing and maintaining the IFC standard. The IFC has also been accepted as an ISO 16739 standard, and with a long history of development behind it, we see it as a suitable choice for a starting point to building information.

4.1.1.1 A Brief History

It all started in 1994 with Autodesk forming an industry consortium which aimed at developing a set of C++ classes which could support integrated development. Twelve US-based companies joined this consortium. By the end of 1995, they had opened membership for all those interested worldwide. This alliance got renamed as the International Alliance for Interoperability (IAI) in 1997 and it was at that time they decided to work towards the development of a vendor-neutral data model, which was called the Industry Foundation Classes (IFC), and which aimed towards serving the entire building lifecycle. This presided with the initial release of the IFC 1.0 standard during the same period. It was quite short lived and a couple of advances followed in the coming years with IFC 1.5 and IFC 2.0. In a release in 1999, the IAI mentioned its vision as "To enable software interoperability in the AEC/FM industry" and its mission as "To define, promote and publish a specification

for sharing data throughout the project lifecycle, globally, across disciplines and across technical applications" [Laakso et al., 2012]. IFC 2.0 was the first comprehensive and truly international standard, and its main scope was to include schemas for building services, cost estimation, and construction planning [Liebich, 2010]. The following period of early 2000's was of a relatively slow growth with minimal industry adoption of IFC. Then with the release of IFC 2x3 in 2005 and its acceptance as an ISO standard, it got a new push. Further, the name of the organisation was changed to buildingSMART International to better represent its purpose, and it still remains.

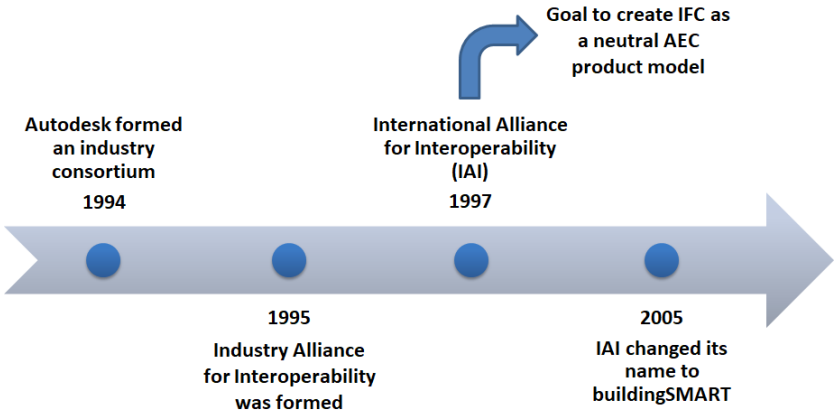


Figure 4.1: History of Organisation

There were many reasons for the slow resonance from the market during the early days, which also included the complex and wholesome structure of the data model, along with little economic motivation. The IFC model was directly adopted from the STEP in terms of the representation of data and the underlying Express schema, which was the reason for its complex structure. There was felt a need to make it more concise and targeted for the needs of industry. Thus, at the same time around 2005-06, the concept of "the useful minimum" was introduced as mentioned by [Hietanen and Lehtinen, 2006]. This facilitated the narrowing down of the data within IFC for exchange purposes, which made it more efficient and adaptable by the industry. An outcome of this was the Information Delivery Manuals (IDM) and the Model View Definitions (MVD). These specifications define what all detail of information is to be included for a particular transaction, and thus, supplement the basic validation formats like Express. The recent developments of IFC 4 and beyond is a proof that this standard is constantly developing with the growing needs and demands within the industry, and it has since been formally accepted even at the level of several national governments.

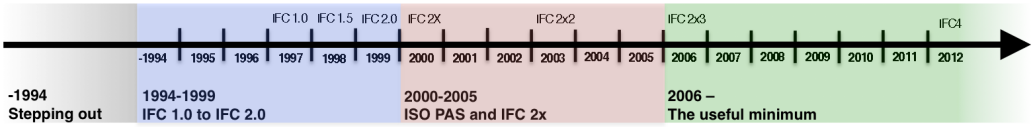


Figure 4.2: Evolution of IFC [Laakso et al., 2012]

4.1.1.2 Architecture and Data Model

We have mentioned earlier that the IFC was directly inherited from the STEP data model, and thus resembles a lot of its features. EXPRESS data definition language is one of these, and with it comes the emphasis on the entities and their relationships within the data model. Although IFC has also been expressed in XSD schema, but it has also been derived in-turn from the EXPRESS, so similar characteristics follow. Thus, we see that the IFC has a huge number of entity definitions, which was around 800 in their latest release, and this leads to increased complexity and difficulty in perception of its structure. In order to ease this process, the official specifications of the IFC release point out to four conceptual layers based on different levels of hierarchy as shown in figure 4.3. The core layer contains all the major generic entities *IfcRoot, IfcProduct, IfcProcess, IfcElement,* etc. and provides a base from where further domain specific, specialised entity definitions can be extended, which then form part of the domain layer. There is another intermediate layer of overlap between these two which caters to general product, process or resource specialisation schemas serving inter-domain exchange of construction information. This would contain definitions like *IfcWall, IfcDoor,* etc. which are further utilised for domain layer entities like *IfcDoorType, IfcDoorLining,* etc. Finally, at the bottom of the hierarchy lies the resource layer which contains all fundamental definitions which are used by other definitions of higher levels. This layer does not include a globally unique identifier for its schema definitions.

Having covered the underlying architecture of the IFC data model, we now discuss how objects and their relationships exist within the IFC structure. The objects inside IFC are arranged in a strict hierarchy, and are assigned various attributes and features based on where they lie within the hierarchy. For instance, *IfcRoot* is the base class of all the entities within IFC except the resource definitions. It has certain attributes like a Global Id which doesn't change during its entire lifetime, Owner History, Name and Description which are passed on to all the classes that inherit from it. Similarly, we have *IfcProduct* which is the base class for all entities representing physical or spatial elements like *IfcSite, IfcBuilding, IfcBeam, IfcWall,* etc. These all sub-classes lie at different levels of hierarchy further within

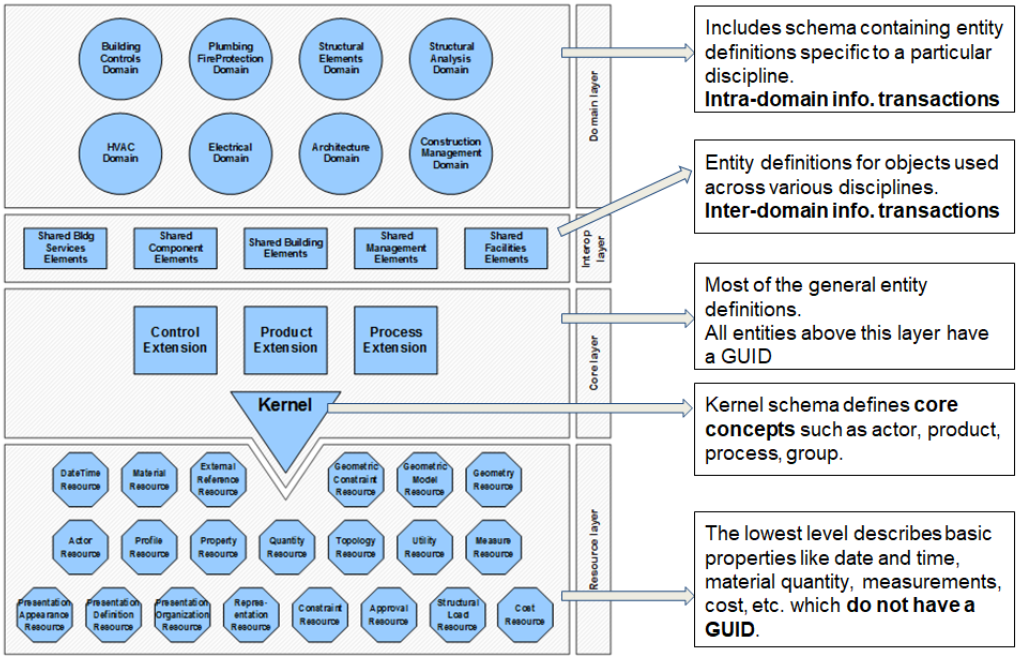


Figure 4.3: IFC Architecture [Recreated from IFC4.1 Official Documentation]

the structure. Unified modelling language (UML) diagrams of *IfcRoot* and *IfcObjectDefinition* classes are shown in figure 4.4, which show the various features and classes inheriting with each of them.

Another interesting thing about the IFC data model is that relationships are first-class objects, i.e., they are represented as standalone entities rather than just connections. This makes the IFC structure much more comprehensive but more complex at the same time. These are inherited from the *IfcRelationship* class and connect to the related elements on both sides.

4.1.1.3 Putting into Context

Given the comprehensive structure and the wide international acceptance of the IFC standard, it is a strong candidate to serve all kinds of building information from geometry to semantics, over the entire lifecycle of the asset. It also ensures the inclusion of all the relevant data that we might require in our application. Although there are other formats like RDF, which was discussed before, which are lighter and much better in certain aspects like querying the data, they do not suite fit for our purpose because of largely excluding the geometric information from the picture, which is the primary

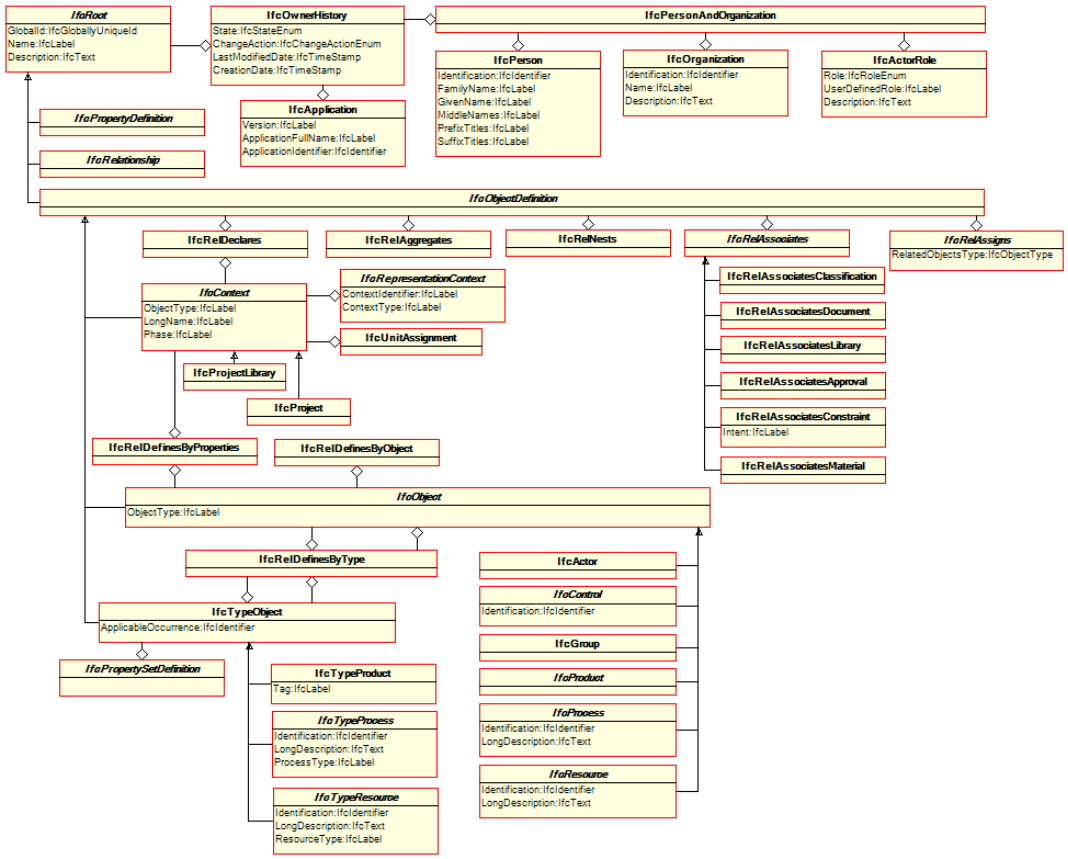


Figure 4.4: UML Diagram for *IfcRoot* and *IfcObjectDefinition* [BuildingSMART-Tech]

requirement for visualisation. Another important consideration is about the tools required to extract visualisable geometry from the building model. IfcOpenShell is an open source software library meant for helping through the development using IFC files, especially the geometry within them. The IFC specification uses a unique identifier for object instances that follows the universal unique identifier standard (UUID) with its implementation as a globally unique identifier (GUID) [BuildingSMART-Tech]. This is a 128-bit number associated with every object within IFC and remains unchanged throughout its lifetime. This GUID can be very useful in reconnecting the elements to their related information and to other elements once we separate them to achieve granularity. All in all, the IFC provides a single monolithic package of information and we see it as a suitable source to start with, in the context of the current effort.

4.1.2 Collaborative Design Activity (COLLADA)

COLLADA is described as an advanced 3D asset description, which mainly provides a format for interchange of 3D data. For easy transmission of information between different applications, it uses an XML-based schema which also ensures minimal loss of information. It provides support for encoding and representing a wide variety of visuals including geometry, animations, kinematics, shading, etc. It has been developed and managed by the non-profit Khronos Group since 2006, although the initial release was made earlier in 2004. It is mainly used in the entertainment industry like games and movies as an effective intermediate format in the content pipeline, and was developed primarily for the same purpose. Although it might also be used in industries other than those mentioned above. Also, it provides a relatively smaller file size along with maintaining all the information as well as meta-information, which further makes it a preferred choice. Here, we try to present a brief description and its relevance to our work.

4.1.2.1 Background and Data Structure

The first attempt to create an interactive computer graphics platform was made in 1963 in the form of Sketchpad by Ivan Sutherland. Although he is now considered as the father of the modern computer graphics, little did he know at that time about the problems faced by developers in the coming years with regards to the interchange of graphics data across platforms. This problem of 3D interchange is a rather old one and there have been ad-hoc efforts made by developers all through these days towards formats for interchange. But it was only after the surfacing of web and the XML in the 1990s that a need for standardising of data exchange formats was felt in the 3D graphics as well. The outcome of one such effort was the COLLADA format.

The COLLADA format was developed mainly keeping in mind the provision for an intermediate format in the interactive gaming industry. Any interactive graphics rendering has two parts to it – the platform which provides the interface and interactivity to the user, and the content itself. In the earlier solutions, these two parts were not separated. But with more advancements in techniques, and content getting bulkier and more complex, these are now handled separately. The content, in-turn needs to be developed and then fed into the renderer. In order for this feeding and the subsequent rendering to be efficient, the content needs to be optimised which requires it to be passed through various stages of processing. This is known as a content pipeline which takes in a raw file from an authoring tool, and optimises these files for fast loading and reduced size. It was as a part of this content pipeline

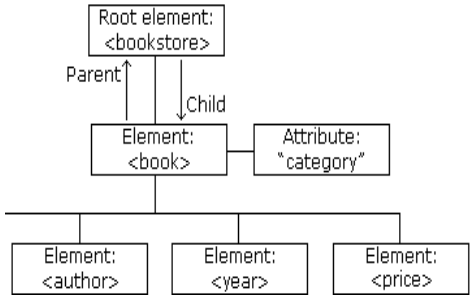
that the COLLADA format was meant to serve as a standard format. And this has been mentioned as one of the design goals among others which were listed as a part of the latest specifications of COLLADA in 2008 [Barnes and Finch, 2008]:

- To liberate digital assets from proprietary binary formats into a well-specified, XML-based, royalty-free, open-standard format.
- To provide a standard common language format so that COLLADA assets can be used directly in existing content tool-chains, and to facilitate this integration.
- To be adopted by as many digital-content users as possible.
- To provide an easy integration mechanism that enables all the data to be available through COLLADA.
- To be a basis for common data exchange among 3D applications.
- To be a catalyst for digital-asset schema design among developers and DCC, hardware, and middleware vendors.

Since the release of the latest specifications of COLLADA, the focus has remained mainly on its integration with the existing platforms which has led to a set of peripheral tools and specifications like the COLLADA Conformance Test Suite (CTS) in 2011. Another notable effort is that of OpenCOLLADA which is community effort led by the Khronos Group for providing open source tools for the COLLADA format and is freely available on GitHub. Naturally, such an efficient data format could not have remained limited to only a single field for which it was developed, and it has been rapidly incorporated for all kinds of purposes related to digital 3D representation like training simulators, movies, architecture, and even GIS. With the years, COLLADA has kept up with the expectations of the users, which is evident from its wide acceptance and support within all kinds of 3D applications and tools.

As far as the schema of the COLLADA format is considered, it is based on the Extensible Markup Language (XML), which was designed to carry data, especially over the web and provides the flexibility for all kinds of data. COLLADA schema defines its own rules for the XML elements, and any file which confirms to those rules is referred to as a Digital Assets Exchange (`.dae`) file. XML describes the content, its structure and semantics through blocks of information enclosed within tags which can be arranged within each other forming a hierarchy (Figure 4.5a). These XML elements might

also have attributes providing metadata, which can then be used to identify them. The COLLADA schema defines syntax for addressing these elements through unique identifiers in the form of URI and Scoped Identifier (SID) (Figure 4.5b). Another feature is the profiles which define the context for representation of information within the format. These profiles are readily understood by the tools which operate using COLLADA. It also establishes the naming conventions for elements within these common profiles. There are a lot of other specifications meant to describe effects related to animations, kinematics, physics, etc. but those are out of scope of the current work, as we will just be dealing with physically static building models.



(a) XML Tree Structure [W3Schools]

```
<channel source="#YFOVSampler" target="Camera01/YFOV"/>
...
<camera id="#Camera01">
  <optics>
    <technique_common>
      <perspective>
        <yfov sid="YFOV">45.0</yfov>
        <aspect_ratio>1.33333</aspect_ratio>
        <znear>1.0</znear>
        <zfar>1000.0</zfar>
```

(b) URI and SID Example in COLLADA

Figure 4.5: Data Representation in COLLADA format

4.1.2.2 Relevance

Having provided an idea about the roots of COLLADA and its data structure, we now proceed to discuss how it is relevant in the context of current work. We required a format to store our geometric data which is retrieved from the IFC files, and which could easily and efficiently render our building models. Having the idea of web-based rendering of building geometry, we looked for something compatible with the web standards. COLLADA provided an XML-based data structure, and having in mind the similar XML-based format of SVG for 2D working good previously, it provided a good choice. Moreover, it provides a small file size despite being XML-based, which further increased its merit. Another positive thing about the XML-based format is that it is content-scalable. We see that it is being readily accepted as a standard format all across the industries related to 3D graphics, and even the AEC industry is not behind on that front. Architecture, CAD and BIM tools like ArchiCAD, Autodesk InfraWorks, BricsCAD, Chief Architect Software, etc. are increasingly providing support for the COLLADA format. Further,

it also caters to our specific need of identifying individual elements on their separation from the whole model by providing an addressing system which stores the GUIDs from the IFC in the form of URIs as attributes inside the COLLADA files (Figure 4.6).

```
<library_visual_scenes>
  <visual_scene id="IfcOpenShell">
    <node id="3zk2ScwerEPudYAqJKNTVo" name="3zk2ScwerEPudYAqJKNTVo" type="NODE">
      <matrix>-1 0 0 10.8 0 -1 0 10 0 0 1 0.95 0 0 0 1</matrix>
      <instance_geometry url="#representation-27992">
        <bind_material>
          <technique_common>
            <instance_material symbol="IfcWindow" target="#IfcWindow"/>
          </technique_common>
```

Figure 4.6: IFC GUID as represented in one of the converted .dae file

4.1.3 GL Transmission Format (glTF)

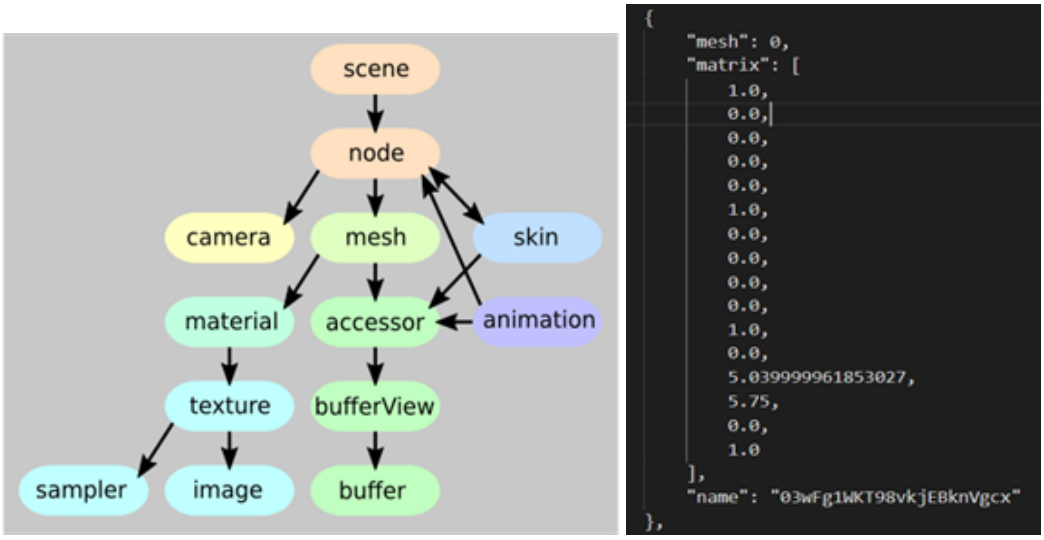
The Graphics Library Transmission Format (glTF) is an open-source, vendor-neutral format for 3D content to be efficiently delivered from the content creation tools to the rendering tools, also referred to as asset delivery. The standard 3D file formats before this were generally focussed on providing collaboration between different platforms or serving in the content pipelines and were mostly optimised for offline processing. But none of them really focussed on optimising downloading and transfers over the web. glTF was aimed to provide a generic model to store, carry and render 3D models without any specific parsing requirements, just based on the capabilities of the GPU of the host device. With its small size and relatively easy rendering, it aimed to fill the gap created by an absence of a simple and common format for 3D scenes and models. This is the reason that it has been described as the "JPEG of 3D" by its creators. It is a relatively new, JSON-based, cross-platform standard and is developed and maintained by US-based, non-profit Khronos Group.

4.1.3.1 Background and Data Structure

The initial idea for a JSON -based 3D modelling format was presented in 2012 at a meeting of Khronos Group discussing about the COLLADA format and opportunities around WebGL. After two years of development process, the initial release of glTF was made in 2015. Around the same time, there was a need felt within the industry to have a standard delivery format for 3D

media, something which can be seen through the words of John Carmack, who in 2016 said that "*The world has long needed an efficient, usable standard for 3D scenes that sits at the level of common image, audio, video, and text formats. Not an authoring format, or necessarily a format you would use for a hyper optimized platform specific application, but something at home on the internet, capable of being directly created and consumed by many different applications*".

The glTF seemed to serve well towards this purpose, which is why the industry giants quickly started to adapt this format, and even contribute towards its further development. It was not long before the next version glTF 2.0 was published in 2017, which introduced a number of new features like Physically-Based Rendering (PBR) materials, and API-neutral rendering. The design goals mentioned as a part of the specifications include provision for compact file sizes, faster loading, runtime-independence, complete 3D scene representation, and extensibility [GLTF 2.0 Specifications].



(a) Scene Structure in GLTF [GLTF 2.0 Specifications]

(b) IFC GUID number stored as 'name' attribute of mesh

Figure 4.7: Data Representation in GLTF

As far as the data structure of glTF is considered, it includes JSON files with optional external supporting data in the form of binary files containing geometry and animation data, and image files containing texture information. The file defines 3D models contained within a scene, and has certain division within its structure which is identified in terms of some top-level elements namely: scenes, nodes, cameras, meshes, buffers, bufferViews, ac-

cessors, materials, textures, images, samplers, skins, and animations. These elements are further connected to each other forming a hierarchical arrangement as depicted in figure 4.7a. Inside the file, these are defined as arrays of data, objects within which may be referenced using their indices.

The scenes and nodes elements define the basic structure of the 3D scene. A scene may contain multiple nodes as its children elements, which may further have other nodes within them, thus forming a hierarchical structure. The nodes can have attributes like "name" assigned to them. This can be useful when we are trying to store our GUIDs and IFC type information associated to the geometrical elements inside the files. These nodes may then define properties for transformations and references to meshes and cameras, which define the rendering and the scene conditions respectively. The mesh defines something known as primitives inside it, each of which defines a rendering mode which is either in terms of Points, Lines or Triangles. It is important to note here that all the geometry is brought to the level of three simple geometries. The graphics cards inside PCs can only render triangles, because this is the only shape that is guaranteed to be geometrically correct in 3D for any set of three points.

Further, the buffers define the actual geometry and are related to the mesh via the accessors and bufferViews, which are responsible for adding structure and layout information to the geometry. There is also other information like materials, animation, skinning, etc. which is attached to various elements. The external or even internal buffer/image resources within the glTF are referenced to by Uniform Resource Identifiers (URIs). The glTF 2.0 also specifies a binary format with the extension of `.glb` which acts as a container element for the JSON, binary buffer and image resources providing a single file with smaller size.

4.1.3.2 Relevance

Coming on to the relevance of the glTF format within the AEC domain, and specifically for our purpose, there are a couple of things to mention. We have already seen how IFC is the best available format for exchange and interoperability of building information models. Its merit largely lies in the fact that it is an all-inclusive format which can represent all kinds of information related to the building lifecycle. This very fact also becomes the cause of its inefficiency when it comes to simple concentrated tasks like just viewing the model. We have seen in the previous section how the glTF format provides a simple way of representing and rendering of 3D scenes and models. Also, the representation is similar to that used inside the graphics processing units of PCs, thus making the rendering further efficient and getting rid of the

parsing requirements. Another important criterion for us was to attach the GUID number to the file, which was achieved through the "name" attribute of the mesh (Figure 4.7b).

Coming to the industry dynamics and acceptance, we have seen in section 2.3.2 how the various solutions available in the market approach the problem of visualisation. Most of their viewers can only display their own proprietary format, and they convert everything into that behind the hood. If we look into the past of digital construction industry, we find that the DWG format dominated the industry at one point, and there was little scope and support for alternative formats because of the monopoly of a major industry player. But times have changed now, and already having the support of major tech giants, glTF format cannot be suppressed by the construction industry players. Also, the construction industry itself has become more aware of the importance of open standards and it would readily accept anything useful which comes its way. We also find that the support in terms of development is continuously increasing for glTF with a wide variety of open-source tools from importers, exporters, converters to validators, loaders and viewing engines. Thus, we see that for the purpose of visualisation, glTF provides an open, neutral, simple and sufficiently supported format with a small size and faster rendering.

4.2 Pre-Processing Tools

As we chose the IFC file to be the single source of building model for our application, there was a need of certain pre-processing to be done on that data in order to implement it and make it useful for our purpose. There were two primary requirements – to extract the geometry out of the IFC model into a visualisable format, and to extract the semantics of the IFC in order that they may be easily queried and connected to the visualisation platform. The main constraints for both of these were that the tools should be easily accessible and applicable. This section introduces the tools which were used in our case and provides the context of their application.

4.2.1 IfcOpenShell

The file formats we described above require certain tools to be converted from one to the other. IfcOpenShell is an open source software library which is meant to facilitate the development using IFC files. It uses Open Cascade to convert the implicit geometry inside the IFC files into explicit visualisable geometry. It provides a range of products including importers for certain

```
~$ IfcConvert input.ifc single_output.dae --include+=arg GlobalId 03wFg1WKT98vkjEBknVgcx
  --use-element-guids
```

Figure 4.8: IfcOpenShell Command Line Code Example

software packages to a stand-alone converter to OBJ format. IfcConvert is one of the prominent solutions it offers, which facilitates conversion of IFC file to a number of 3D and 2D geometry formats including Wavefront OBJ (`.obj`), Collada (`.dae`), STEP (`.stp`), IGES (`.igs`), XML (`.xml`), SVG (`.svg`). It provides a command line tool to execute these conversions and also provides a number of advanced features which serve specific purposes.

We had identified to use Collada format initially for storing the explicit geometries because of various reasons which have been mentioned earlier and which will also be covered in detail in the next chapter. Considering that choice, and the method in which we wanted to convert the geometry, i.e., element-wise conversion identified by their GUIDs, IfcConvert provided a good option as we could restrict the conversion to individual elements using its "–include" option and providing the GUID number as the argument (Figure 4.8). It has been originally written in C++ programming language, and has a slow development history because of limited industry support. But that is counter-balanced by an enthusiastic community support. But above all the factors, the granular access to the IFC geometry makes it our preferred choice.

4.2.2 Collada2GLTF Converter

Once we started implementing Collada in our renderer, we came across glTF format, which was more effective both in terms of size as well as rendering. But the problem was again how to convert granularly element-wise from IFC to glTF. In absence of such a tool, we then followed a two-step process and converted the Collada files to the glTF format using the COLLADA2GLTF converter provided in public domain by the Khronos Group, an organisation which is also responsible for the development of both these formats in the first place. The discussion on glTF format has already preceded, and considering those merits, this was the most suitable tool for the application.

4.2.3 IFC2Graph Converter

When we consider converting the semantic data within the IFC into a graph based data structure, we find a lot of different options in terms of the tools.

The IFC2Graph Converter is a tool developed previously as a part of this project, and had been implemented for a granular 2D platform by Zibion [2018]. It is written in Python programming language, and employs libraries like IfcOpenShell, PythonOCC, and Py2neo. The objects within the IFC files are converted to nodes inside a graph database. The graph database used by this tool is Neo4j, which is an open source native graph database platform, further discussion on which follows in the coming sections.

As we have already seen, the properties within the IFC may be represented as separate entities, which are related indirectly to the objects. Thus, getting properties of the objects and storing them as attributes directly related to the nodes was an important step towards simplification of the data structure, and in-turn improving the querying efficiency. Similarly, the relationships between objects have a very complex representation within the IFC, which is again simplified in the graph database. Multiple independent efforts like Khalili and Chua [2013], Ismail et al. [2017], etc. have been made to model IFC data as a graph, and even the discussion of linked building data and semantic web surrounds this concept. But we stick to the custom made converter for this effort, as it is just a proof of the concept implementation. And further improvements might be made based on detailed analysis, as has already been mentioned initially. For this prototype implementation, the Neo4j graph database serves well, and so does the converter implemented for it.

4.3 Web Development Tools

The internet has really become an integral part of our lives in this day and age. We have seen a completely new realm of industries come up in the past couple of decades solely based on the widespread use of the internet. And even the existing industries have adopted the growing technologies leading to a drastic change in the manner of how things work. Within all this, the construction sector has still largely stuck to the traditional methods of how things are done, and has been quite reprehensive in adopting the newer technologies. Although lately they have begun to identify the benefits and the pace of adoption is increasing. With such an increasing and efficient reach of the internet, it is really important to capitalise on its benefits and provide such solutions which are easily accessible and at the same time resource effective.

The World Wide Web as we know it today is built up on a combination of several different tools, which are primarily concerned with data, its representation and interaction. These three tasks are handled by HTML, CSS

and JavaScript respectively in the traditional web development environment. Further, there is a division of content and technology based on what side of the application they are serving. There are some things which interact with the user and take care of what goes on the client-side. The other side to it is the back end which takes care of the database and related stuff which is detached from the presentation of data to the client or its interaction. This is also known as the server-side of the application. Our main aim is to incorporate better and granular visualisation for the user which has more to do with the front end development. JavaScript is primarily a front end scripting language, and will be our main tool for the purpose. In this section, we discuss the primary tools used for the development of our web-based prototype application.

4.3.1 Web 3D Rendering and Associated Technologies

4.3.1.1 Background

JavaScript is an imperative programming language, which means it is used to change the state of a program. It is in-fact the element which makes the static HTML pages to interact with the user and hence, making them dynamic. Further, with technologies like Ajax, it can even retrieve external data through web, and can connect the front end to the server side thus, achieving a truly dynamic nature. It can access and effect changes in the Document Object Mode (DOM), which is an application programming interface attached to the HTML document and which identifies the structure of the HTML document.

The idea of 3D rendering on web has been there for some time, and can be traced back to the first international conference on the World Wide Web in 1994, which led to the development of the Virtual Reality Markup Language (VRML). It was succeeded by X3D in 2003, which had a more modular form and provided better efficiency. But there was a huge shift in the way we looked at 3D web rendering in 2011 with the release of Web Graphics Library (WebGL), which provided an API for 2D/3D rendering without the use of plug-ins. Before that, browsers needed external pug-ins to render graphics. But it provided a way to use the built-in graphics card of the PC for rendering in the browsers.

4.3.1.2 WebGL

According to the Khronos Group, which is the developer of WebGL, it has been described as “a cross-platform, royalty-free web standard for a low-

level 3D graphics API based on OpenGL ES, exposed to ECMAScript via the HTML5 Canvas element". In simple words, it allows browser neutral rendering of 3D graphics on web, depending only on the computer hardware rather than any external plug-ins. Its development started in early 2009 and the latest version has been released in 2017. It has been designed as a rendering context for the HTML5 canvas element. Thus it can interact with other HTML elements as a part of the DOM structure (Figure 4.9).

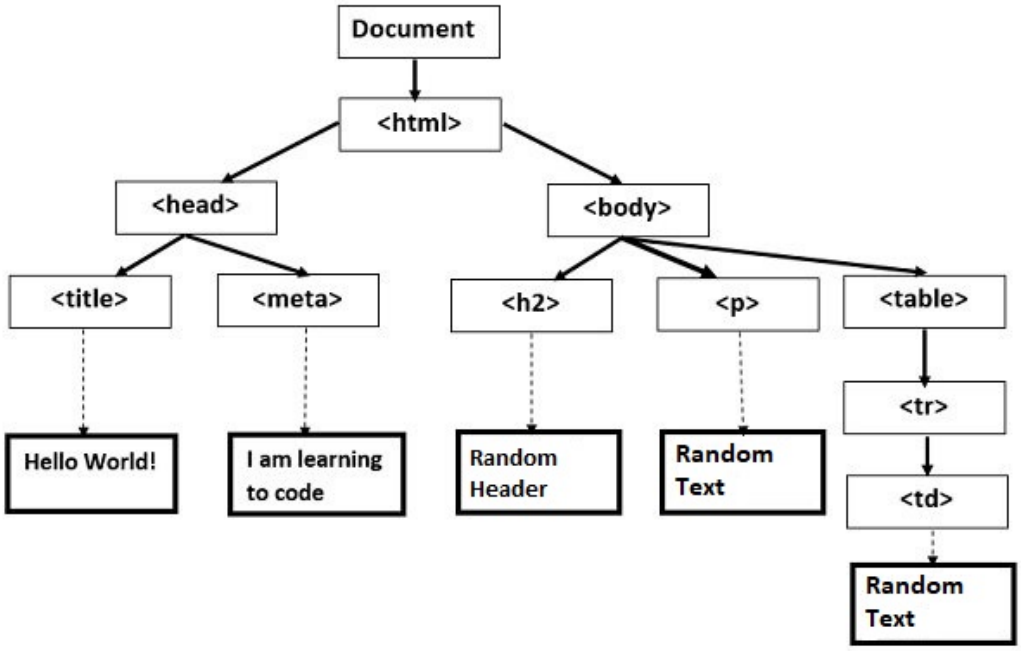


Figure 4.9: Example of DOM Structure in an HTML document

Another important point to mention here is that WebGL is an immediate mode rendering API, which means that the application is in direct control of the drawing commands on the user interface (UI). As opposed to this concept, retained mode APIs duplicate the application state in the graphics library which is then in control of the UI changes, making the process memory intensive. Although the retained mode provides an easier and faster development process, immediate mode provided a better alternative in our case as it involves loading dynamic elements.

WebGL provides flexible primitives which make it easy to develop further specialised APIs on top of it, thus easing the development process. There have been a number of high-level APIs built using WebGL like BabylonJS, PlayCanvas, three.js, etc. which have provided frameworks facilitating utilities ranging from basic tasks like scene loading to more advanced features.

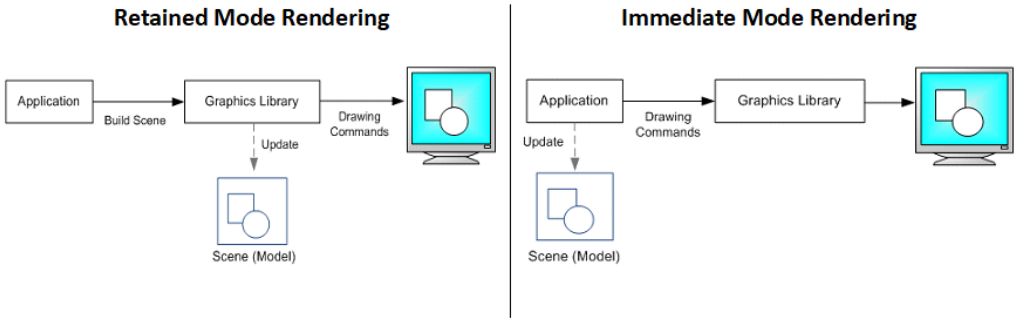


Figure 4.10: Retained vs Immediate Mode Rendering [Microsoft Windows Graphics Documentation, 2018]

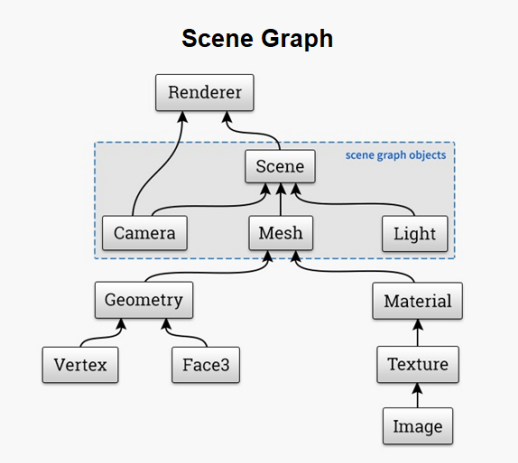
Apart from the rendering utilities, there have been game engines and content creation tools like Blender, Autodesk Maya have actively supported the WebGL API. We have used three.js high-level API for the current work, the details of which follow next.

4.3.1.3 Three.js

Three.js is a JavaScript library and a high-level API based upon WebGL which is used to display 3D computer graphics and animations in a web browser. It is written in JavaScript language and is available as a single file, which may be included in the webpage via a link to either a local or an external source. It allows for rendering 3D content inside a browser using just the capabilities of the graphics processing unit (GPU) of the computer, in-turn making it independent of external plug-ins and/or applications. Its initial release was made in 2010 on GitHub, and it has had a significant number (1071 as on 24/04/2019) [GitHub - Three.js] of contributors, with Ricardo Cabello being the main person behind it.

The basic structure of a three.js application defines a hierarchy of elements as shown in figure 4.11a. The main components include a scene, renderer and camera which are the only necessary elements to display any content (Figure 4.11b). The scene contains all the other elements defining the content within itself. These may include, but are not limited to lights, controls, mesh, external models, etc. The mesh further defines the geometry, material, and other things related to particular objects which are part of the scene. All the three.js objects are loaded onto a canvas element of the HTML5, which defines its position on the webpage using the DOM.

Being dependent only on the browser and GPU, there are bound to be some limitations in this technique of using WebGL/Three.js. One of these is related to memory restrictions. Large models cannot be loaded because



(a) Scene Structure [Lyons, 2014]

```
var scene = new THREE.Scene();

var camera = new THREE.PerspectiveCamera( 75,
window.innerWidth / window.innerHeight, 0.1, 1000 );

var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth,
window.innerHeight );
document.body.appendChild( renderer.domElement );
```

(b) Scene Definition

Figure 4.11: Three.js Scene

of the limited amount of browser cache, which can't be exceeded and might crash the application in case massive datasets are attempted to load. Another drawback is in terms of support for older systems and browsers, which do not have dedicated GPUs and support for WebGL technology respectively. But this can still be overlooked as use of older systems is very limited and is constantly being replaced with newer software and hardware infrastructure. The solution to the former problem has been discussed in later chapters.

Despite some limitations, three.js has increasingly become the world's most popular framework for 3D rendering on web using JavaScript and is being used in a wide variety of applications including gaming, entertainment, scientific data visualisations, model viewing, architecture, and much more. Further, it provides the access to 3D visualisation from computers to mobile devices, thus making it reach far wide. All in all, with an active community and industry support both in terms of its development and use, it forms a great choice for the case of BIM visualisation which has also been previously verified by researchers in this field like Shojaei et al. [2015] for their effort of 3D cadastral visualisation.

4.3.2 React.js

An important part of any application is the user interface (UI). After all, that is the only thing which the user will ever experience. While an intuitive, smooth and fast UI would be extremely good from the point of view of a user, it can prove to be an equally difficult challenge for the devel-

oper. React is meant for this very purpose of easing the process of building better UIs. It is a JavaScript library for building UIs, and is maintained by Facebook along with community support. It was developed by Jordan Walker at Facebook with the initial release being made in May, 2013. React is mainly focussed on developing single-page applications (SPA), as opposed to the traditional multi-page approach. We will start by explaining the basic client-server model followed by a short discussion on the SPA, which would further lead us to a better understanding of React.

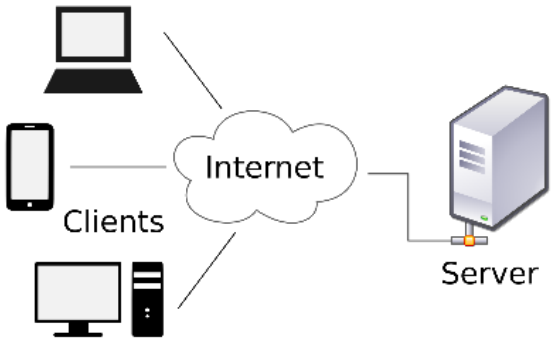


Figure 4.12: Clients communicating with a server via the internet [Client–Server Model, 2019]

The client-server model is a distributed application structure, which means it partitions the tasks of a system between the resource providers (or servers) and the service requesters (or clients), who communicate over a network to get the task done [Client–Server Model, 2019]. This is the model on which the World Wide Web currently works (Figure 4.12). The same can be seen in figure 4.13, if the user clicks on the about link on the index page, a fresh request is sent to the server which then returns the about page. In case of React which works on SPA, it takes over the connection and serves the user with the component one wants to view, without requiring to request the server or loading the page again.

In the traditional multi-page approach, every time there is a need for an update, the client sends a new request to the server, which returns the entire page all over again, thus requiring a reload. The pages are represented in HTML, with CSS styling it and JavaScript serving the interactions. Some of the problems of this approach include loading the same page again and again, no real-time updates, and a complex server-side logic. The SPA approach has shifted most of the logic on the client side, thus allowing more efficient and targeted back-end codes. Now the website interacts dynamically with the server, while updating the pages without reloading them all together. In

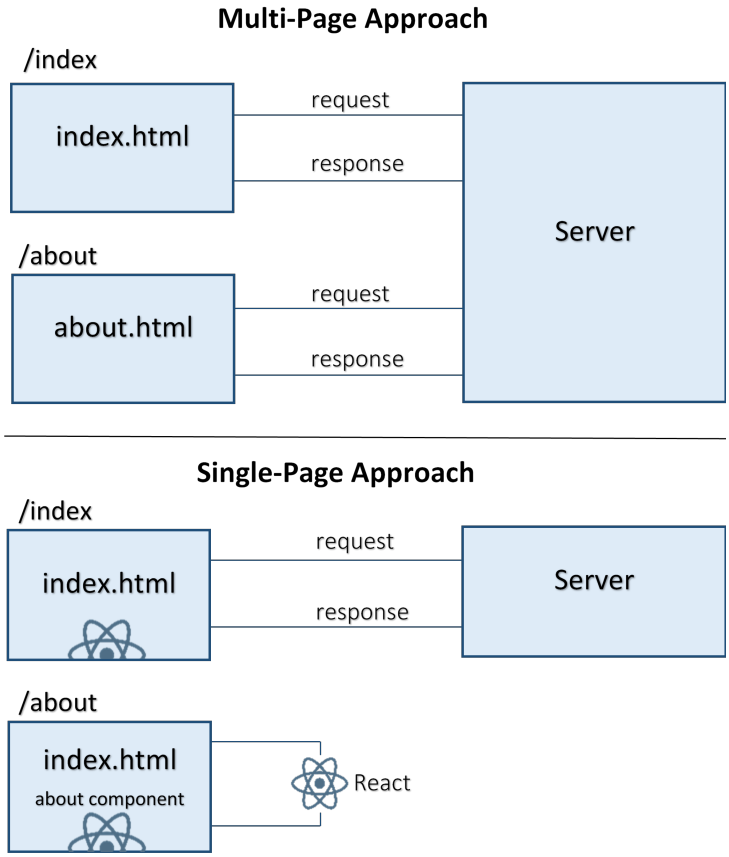


Figure 4.13: Example of Server Connections in Multi-Page vs Single-Page Applications

the SPA world, JavaScript plays a central role in controlling both the other components – data (HTML) and styling (CSS). It also gives the user the feel of a desktop application because of uninterrupted interactions, and is definitely useful for mobile application development. But at the same time, the computations on the client-side have become more complex, leading to the development of front-end JavaScript libraries like React, Vue, Angular, etc.

With a number of distinctive features tailored to help through the development process, React is a widely accepted and growing front-end technology. Instead of separating the mark-up from the logic, it actually combines them into bundled units or components which serve particular parts of the UI. These components maintain a state which can store data and also pass it on to their children components through props. Each of these components implements certain life-cycle methods, which control the execution of code

during certain stages of its lifetime. 'Render' is the most important lifecycle method and the only one which is required. It contains the information for what should be displayed for that component. All this is achieved through the use of JSX, which is an extension of JavaScript to include HTML-like tags within the logic.

Along with the component-based structure, React also implements something called Virtual DOM (Figure 4.14). This has more to do with optimising the network requests and thus, increasing speed and smoothness of the web application. React maintains its own copy of the DOM which is rendered with the initial load. After this, if any updating is required, it does not send the request directly to the server. Rather, it compares the changes required with its own copy and only updates the required elements, thus making it work faster.

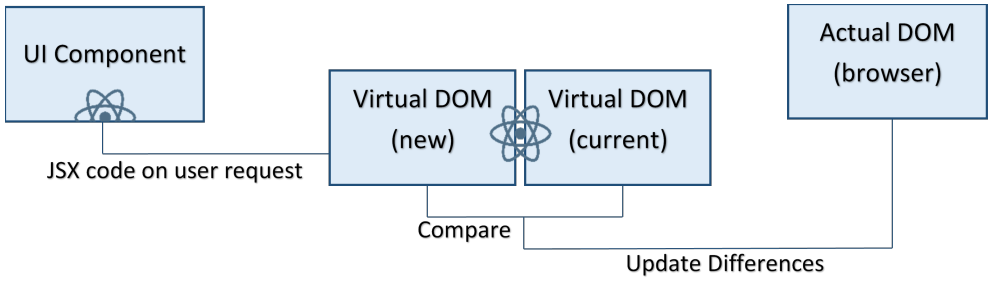


Figure 4.14: Working of Virtual DOM

React mainly aims to provide a front-end development tool to build better UIs and control the browser flow. It is a popular framework with active support which makes it a good choice for development. For our case, we use Three.js for rendering partial 3D building models, and embed it within a React.js framework to create a complete and wholesome user interface for our application. Since we are not focussed on developing back-end architecture for this case, React further appeals us by detaching itself from any kind of back-end. Also, we plan on integrating granular 2D floor plans which have been built upon React framework (due to ease of using SVG within React) with our application. This further increases its merit as a preferred choice. Although there are some low points of React in terms of high memory requirements (because of its use of Virtual DOM), and one-way data flow inside components, but these are far superseded by its numerous benefits.

4.4 Graph Database

Until now we have been talking about the file formats, pre-processing, and front-end web development tools. This section introduces the concept of Graph Database, specifically Neo4j, as the main data storage for the semantics of our IFC model. This serves as our back-end infrastructure for communicating with the webpage which is being run on React.js. Along with that, we discuss about the Cypher query language which is used to query data from Neo4j graph, while finally focussing on the Neo4j driver for JavaScript which actually enables the connection between our JavaScript code and the graph database.

Relational Databases have been the predominant solutions for data storage available in the market since many years. But they have been problematic in adapting to highly connected data due to their inability to efficiently model relationships. The departure from relational datasets has been tried since 1960s, but it surged in the early 21st century with the NoSQL databases coming into picture. However, even these failed to serve the connectedness of data natively. Rather, it required further processing at application level to access the connections as these were represented implicitly. Opposed to this, the graph databases natively serve the relationships as first-class citizens, and make way for two-way connections which facilitate much more extensive and interesting queries. Thus, we see in the last decade, property graph databases such as Neo4j, JanusGraph, and Sparksee have become more widespread in industry and academia [Larriba-Pey et al., 2014].

The concept of storing data in a graph structure is a new one, but it is deeply rooted in our world. Unlike the tabular relational databases, the real world is rich and interconnected, and displays both uniformity and irregularity in different places. For example, we can see graphs in social, economic, political, planning, transport networks, and many other fronts. Even the social web giants like Facebook and Twitter are now using graph based data structures to represent their enormous datasets. Thus, modelling such data in the form of a graph appears to be a strong concept.

Coming to the construction industry, we see that the IFC data model has a highly interconnected nature and can very well be modelled as a graph. There have been some efforts on this as mentioned in section 2.2.3, like Ismail et al. [2017] who have tried to develop an ontology to translate the IFC semantics into a graph structure. Another, much larger effort is being made by the Linked Building Data Community Group, which aims at connecting the distributed BIM data from various sources through a semantic web platform in an open network called the web of data. It touches on the future

of this technology moving towards concepts like IoT and its applications in Smart Cities. In our case, we use the Neo4j graph to export data from IFC files of various disciplines into a single graph, which can then be utilised along with the visualisation framework.

It is also important to note that graph databases do not mean to replace the existing relational databases, which work really well for many situations. But rather they intend to compliment them for the cases where relational model becomes difficult or inefficient to use.

4.4.1 Neo4j

Graph databases belong to the family of NoSQL databases, with the difference being in the representation of relations explicitly in case of graphs. A graph is basically a representation in terms of vertices (nodes) and lines (relationships) which connect those vertices. A number of graph database solutions have surfaced in the past years, and each has its own characteristics. Fernandes and Bernardino [2018] have compared five of these, and they suggest Neo4j as the best option. Further, Neo4j has been consistently ranked highest in terms of popularity by DB-Engines [2019].

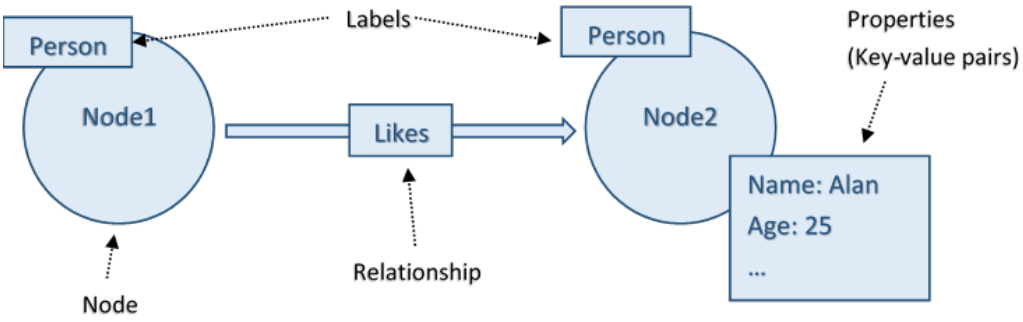


Figure 4.15: Labelled Property Graph Model

Neo4j is a native graph database, which means that it is optimised for storing and managing graphs, unlike some other graph databases which serialise the graph to a relational or some other database. It employs the property labelled graph model, which is the most popular form, others being hypergraphs and triples – the latter being used extensively for the efforts related to semantic web and linked building data. The labelled property graph model has the following properties (as defined by Robinson et al. [2015]:

- It contains nodes and relationships.

- Nodes contain properties (key-value pairs), and can be labelled with one or more labels.
- Relationships are named and directed and always have a start and end node, and can also contain properties.

Neo4j was initially released in 2007, is written in Java programming language, and is available under a GPL-3 licensed open source community edition, apart from a commercial enterprise version as well. It is described as providing Atomic, Consistent, Isolated, and Durable (ACID) compliant transactions, which prevent loss during situations like power or network breakdown and is really important in places like financial systems, etc.

Further, having a graph structure lets it isolate certain sub-graphs out of the entire graph in order to localise querying for specific tasks, thus providing scalability and serving even huge databases with equal efficiency. This ease of traversing nodes and relationships sets the graph model apart from any other type of databases. Once it is setup on any platform, local or on cloud, it can be easily accessed using HTTP calls, which are facilitated by the drivers provided by Neo4j for working with different platforms like JavaScript, Python, etc. Another useful feature of Neo4j is that it provides a built-in browser platform which can be run on any port and which provides documentation, tutorials, templates and many more stuff to learn as you develop (Figure 4.16), and also facilitates querying using Cypher language.

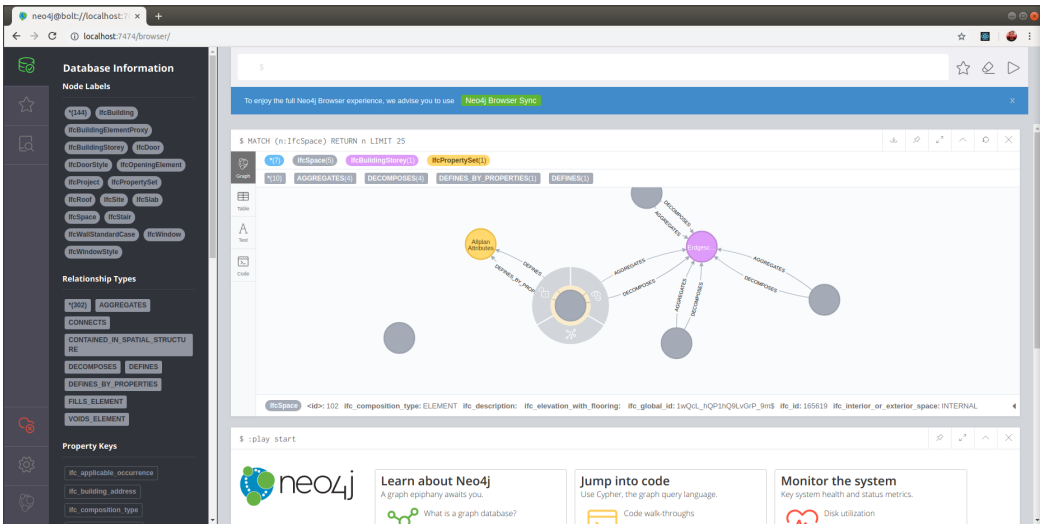


Figure 4.16: Neo4j Browser Interface

4.4.2 Cypher

Once we have populated the database, we need to run queries and access relevant information to utilise in our prototype. Neo4j provides a query language called Cypher which is very good at representing and describing graphs. It is a declarative and a relatively simple graph query language, and has the capacity to handle even huge databases with its compact queries. We have mentioned already that graphs are made of nodes and relationships, but another layer of hierarchy in between consists of patterns. Patterns express simple or complex traversals or paths within the graph and their recognition by cypher works similar to human brain's pattern recognition abilities. Cypher syntax is very similar to natural English language constructs which makes it easily readable and understandable by the user.

Cypher was initially developed in 2011 by Andrés Taylor at Neo4j, but it was made open only in 2015 with the openCypher project. It was designed to be easily read and understood by the developers, database professionals, and business stakeholders, which was facilitated by the intuitive description of graphs using diagrams [Robinson et al., 2015]. Cypher is quite similar to its SQL counterpart to make it easy for experts to switch over. There are also certain other query languages like SPARQL for RDF triples and the imperative language, Gremlin which are being used but we stick with Cypher as it is declarative in nature, and complements the Neo4j graph.

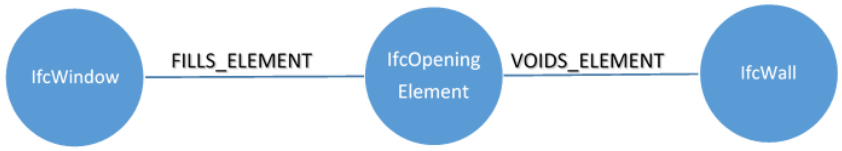


Figure 4.17: ASCII art graph representation of IFC relationship

Moving on to the q of IFC Relationshipuery structure, Cypher has various clauses the most common of which are `MATCH`, `WHERE` and `RETURN`. `MATCH` is used to determine the anchor points or the sub-graphs, patterns around which are then filtered using `WHERE` clause, and finally the data to be sent back is identified using `RETURN` clause. ASCII art graph patterns (Figure 4.17) are fundamental in this process for Cypher. In the example below (Figure 4.18), a simple Cypher query for finding friends of friends is illustrated. We see that Cypher looks for the node labelled person and filters it further using

the name '*Alan*'. It then returns the friends of friends of the person named Alan using the `KNOWS` relationship.

```
$ MATCH (a:Person)-[:KNOWS]->(b)-[:KNOWS]->(c)
  WHERE a.name = 'Alan'
  RETURN c
```

Figure 4.18: Example Cypher Query

We see that cypher provides a good option for querying semantic building data from the graph database in our prototype. We have only discussed about the read operations, but it has much more capabilities like changing the state of the graph, providing several more clauses to serve a wide range of querying operations, but that is out of the scope of this thesis. The writing to the graph is already taken care by the IFC2Graph converter in our case, and we only need to read the data from the graph.

4.4.3 Neo4j JavaScript Driver

Coming to one of the last technical tools we will be discussing, the Neo4j JavaScript driver is the API which provides access of the Neo4j graph database to our application. It facilitates the connection from the web platform (which is based on JavaScript) and the remote database via simple HTTP calls. It helps take Cypher queries from the application based on the user requirements, and serves the returned data back to the application. An example of how it works inside the JavaScript language is shown in figure 4.19.

```
var neo4j = require('neo4j-driver').v1;

var driver = neo4j.driver("bolt://localhost", neo4j.auth.basic("user", "password"));
var session = driver.session();

session
  .run("MATCH(n: IfcDoor) Return n.ifc_global_id")
  .then( function (result) {
    const guid_array = result.record.get(0);
    session.close();
    driver.close();
  });
```

Figure 4.19: Using Neo4j JavaScript Driver

The working of this driver can be explained as sequence of simple steps. We first ask the database for a new driver, which in-turn provides us a new

session. Then the queries can be run through that session, which would return an object representing the result. This result is in the form of an ES6 promise in our case (for JavaScript) which can be accessed via '.then' keyword. Once the results are processed, the session and subsequently the driver can be closed. The example above returns the list of GUID numbers (stored as `ifc_global_id` attribute inside the graph) of all the *IfcDoor* elements inside the graph database which is hosted on a local server (in this example), which can be accessed via the driver authenticating using the username and password set for the database.

Another important and fairly obvious thing to use the driver is that it must first be included in our web application. It can either directly be embedded in the HTML as an external link, which is generally not recommended in case of managing dependencies as the browser might not understand the import statement. Solution to this is to install it as a node module via node package manager (npm). The discussion about node and npm is beyond the scope of this thesis. We will only mention that it adds the driver to our application in a more wholesome way saving us from any dependency problems, especially when using a framework like React.js, which is our case.

Chapter 5

Prototype Implementation

This chapter would be devoted to explaining the efforts for implementation of the prototype application. This has been developed as a proof of concept for the solution approach mentioned previously. Different processes followed, choices made, challenges faced and efforts for their resolution have been highlighted. We start with the pre-processing required on our basic model, moving on to the rendering strategy. Then, we discuss about the intractability of the rendered model and its connection with the semantic data, followed with the discussion of integration with 2D system.

Figure 5.1 shows the overall schema for the workflow of the prototype. We have three separate processes, one each for pre-processing of geometric and semantic data, and another for the application flow. These are then connected to each other through various stages in order to serve the larger purpose of the prototype application. Detailed discussion about each process has been taken up in the following sub-sections.

5.1 Pre-Processing

This sub-section discusses the steps which were required to be taken before we could utilise the data for actual rendering. The concept of modularising building information, which is an important part of this thesis starts off implementing from here.

5.1.1 Converting Implicit Geometry to Explicit

The geometric information contained within the IFC file is implicit in nature, which means that it cannot be directly utilised for visualisation. Thus, it needs to be converted into a format which is readable by any software package

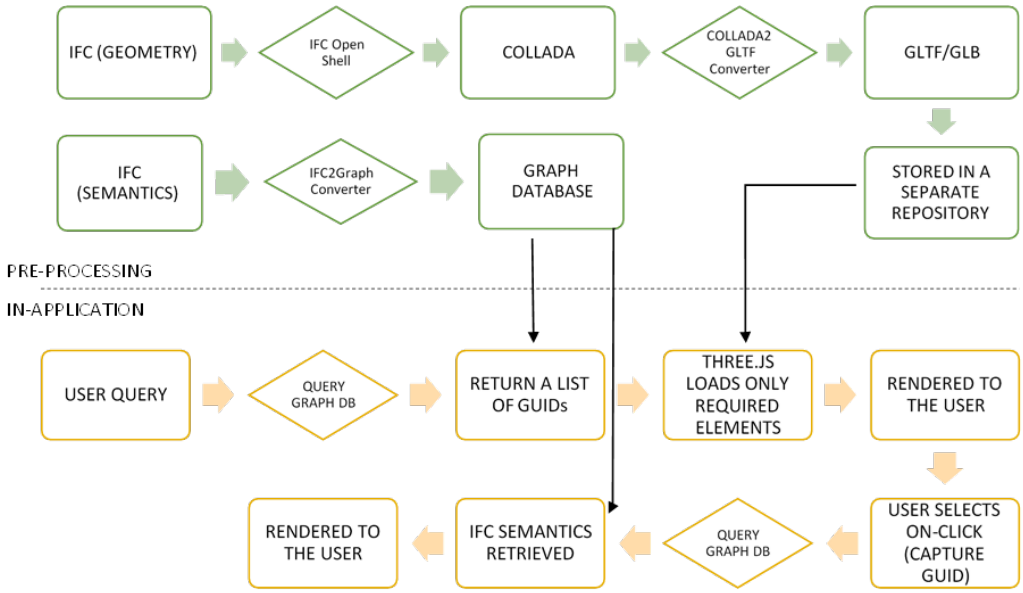


Figure 5.1: Overall Workflow of the Prototype

for visualisation.

5.1.1.1 Choice of File Formats

The first thing which comes to mind when we think about converting the geometric information within IFC file is to what format should it be converted. There are a number of formats which can be utilised to store and later render explicit 3D geometric information. A parametric comparison has been presented in Table 5.1. Each format has its distinct features which can be helpful in many ways. We select the one which suits best for our purpose. The main factors to consider while choosing the best format were that it should be:

- Free & Open Source
- Small in size
- Simple and fast rendering support
- Easily convertible from IFC
- Sufficiently well documented
- Good community support for development

We compared the available formats. The proprietary file formats were dismissed because it goes against the idea of open data. From the available

File Format	Encoding Geometry		Neutrality	Representation	Texture Mapping	Scene Information	Industry Use
	Approx.	Precise					
STL	Triangular Mesh		Open Source	ASCII & Binary	No	No	3D Printing, CAD
OBJ	Polygonal Mesh	Smooth Curves, NURBS	Open Source	ASCII	Yes, but stored in separate file	No	3D Graphics, 3D Printing
			Proprietary	Binary			
COLLADA	Linear Spline	Cubic B-Spline	Open Source	XML	Yes	Yes	Films, Video Games
VRML/X3D	Polygonal Mesh	NURBS	Open Source	XML	Yes	Yes	Virtual Reality
IGES		Precise Mesh, CSG	Open Source	ASCII	No	No	Defence
STEP		Precise Mesh, CSG	Open Source	ASCII	Yes	No	Engineering, Construction
GLTF	Triangular Mesh		Open Source	JSON, Binary	Yes	Yes	Information exchange, Games
FBX			Proprietary	ASCII, Binary	Yes	No	Video games, Films
DWG			Proprietary	Binary			CAD
3DS	Triangular Mesh	NURBS	Proprietary	Binary	Yes	Yes	Architecture, Engineering

Table 5.1: Comparison of Available 3D File Formats

open source formats, we considered file size as a primary factor. This was done because of memory limitations in web-based rendering due browser, bandwidth, etc. We needed to render multiples (in the order of hundreds) of individual element files, so their size was a main concern. Another issue with size is that as the approximation of geometry increases with reducing size. But sufficiently approximate formats are still good for building visualisation, as this in itself is not being used for critical calculations. But at the same time, there were other key concerns like proper support for rendering in a web environment. Many formats cleared this condition like OBJ, Collada, glTF, STL, etc. During the initial selection, another thought in our mind was about the scheme of representation. Although binary provides a smaller size, we initially considered using XML-based format for its likeness to the HTML DOM structure, with the hope that it would help in granular rendering, as was seen in the case of 2D rendering with SVG format. Thus, the Collada format was chosen initially as it provided a relatively smaller size among XML-based formats, along with satisfying the other conditions.

While implementing Collada, however, the XML-based structure was not accessible due to hooding by HTML5 canvas element within the browser. Thus, we went for further smaller sized formats. glTF proved to be a good option, something which has recently developed and grown, and is being supported by major web platforms like Facebook and Microsoft. This is because it minimises both the size of the 3D assets as well as the processing

time to unpack and utilise them, as mentioned by its developers. It provides both JSON as well as binary encoding. We tried both, but finally used binary format which is represented as `.glb`. The reason was the smaller file size of `.glb` with respect to `.glTF`, which reduced at least by about 25% (Table 5.2). The time taken for conversion was also reduced, though not that significantly. Also, most of the open formats like Collada, glTF, OBJ, or STL have dedicated loaders available for use with rendering engines, which provided us with the freedom to choose.

S. No.	IFC (MB)	DAE (MB)	GLTF (MB)	Reduction (% w.r.t. DAE)	GLB (MB)	Reduction (% w.r.t. DAE)	Reduction (% w.r.t. GLTF)
1	41.9	75.6	26.0	65.6	16.8	77.8	35.4
2	–	28.0	10.6	62.1	7.60	72.9	28.3
3	–	8.90	2.20	75.3	1.60	82.0	27.3
4	–	3.40	.496	85.4	.371	89.1	25.2
5	–	.872	.201	76.9	.149	82.9	25.9
6	–	.296	.110	62.8	.081	72.6	26.4
7	–	.219	.075	65.8	.055	74.9	26.7
8	–	.014	.011	23.8	.007	52.4	37.6

Table 5.2: File Size Comparison for Different Formats

Note in the above table 5.2 that there is no IFC file size for the partial elements. Although it would be interesting to note the comparison between IFC file size and the sum of all partial file sizes. Another observation is that the size of DAE file is more than that of IFC when converting the entire model. This is because of the presence of XML-based schema in DAE/Collada which represents the geometry explicitly as compared to the implicit representation in IFC. This data has been collected by generating objects of differing range of file sizes to see the overall pattern of changes. A significant reduction in file sizes occurs from DAE to GLTF, which further shrinks down in the binary variant, i.e, GLB.

5.1.1.2 IFC to Collada Conversion

One of the reasons for choosing Collada initially was also the ease with which it can be converted from IFC. IfcOpenShell library has already been discussed in section 4.2. It provided easy command line interface to convert IFC into multiple Collada files based on the GUID numbers (Figure 4.8). It was finally used along with the other tool in a small pipeline implementation, as explained next.

5.1.1.3 Collada to glTF Conversion

After identifying glTF as our preferred format, the concern was about how to convert each element into a separate glTF binary file. Direct conversion tools from IFC were not available, so the already existing Collada files were used inside the Collada2GLTF Converter, which has been discussed in section 4.2. Another benefit of this was that it could convert directly to binary format `.glb`, with the `-b` option added to the command (Figure 5.2).

```
~$ COLLADA2GLTF-bin guid1.dae guid1.glb -b
```

Figure 5.2: COLLADA2GLTF - From DAE to GLTF/GLB for each GUID

Both this, and IfcConvert were used inside a simple pipeline code in Python language to execute it for the entire model in one go. The resulting files were store in a local repository, from where they could be accessed by the web application.

5.1.2 Converting IFC Semantics to Neo4j Graph

Many options have come up in the recent years when it comes to converting IFC semantics into a graph data structure. These efforts have been discussed in detail in section 2.2. Being independent efforts, the availability of the code from these efforts for our purpose was an issue. Many of these approaches have great potential, but for this work, we stick to Neo4j graph database which provides a labelled property graph model. The conversion was implemented by Zibion [2018], and the code was available to us from the author. It takes in an IFC file and converts the entities and their relationships into corresponding nodes and relations inside the graph. The geometric data is not translated. A benefit was that multiple IFC files could be translated into a single Neo4j graph which would later help to provide inter-domain data interactions.

The graph was currently hosted on a local server using Neo4j desktop application, but would be later be transferred to a cloud server for better accessibility and performance. The converter works in a python environment which was set up on a Virtual Machine using Linux operating system. The same was also used for the initial set-up of the graph database.

5.2 Rendering

5.2.1 Mode of Rendering

Rendering the models to the users was another important step, and the mode of rendering is critical when looking at improving the extent to which it is used. For example, if the designer working in his office, and the supervisor working on the field would have different requirements and capabilities in terms of infrastructure, etc. A heavy software can be used inside an office, for example, but not on hand held devices on site. Internet has brought the world to our hands today, and with the advent of 3D graphical rendering over the web, its reach in related professions has increased. Thus, when we look for providing a granular access to visualisation, there is no better option with a wider reach than the web. Then we look at the direction in which the building data infrastructure is developing. We find that it is rapidly endorsing the idea of semantic web and linked building data, and efforts have even been to represent 3D geometry over the web [Pauwels et al., 2011]. It is highly likely that all data transactions in the near future would be happening over the web. Hence, we chose web-based rendering as the mode of our visualisation.

5.2.2 Choice of Renderer

Web3D is the term used to identify any kind of interactive 3D technology used over the web. It may include file formats, APIs, game engines, etc. Conventionally, external browser plug-ins were required to use 3D technology in a web environment. This has now phased out with the maturity of the technologies like HTML5 canvas element, and WebGL, which allow plug-in free 3D rendering inside web browsers, using only the capabilities of the GPU on the computers. This is why WebGL has been widely accepted and implemented by all major browsers, who have in-fact also contributed to its development in the first place [Jackson, 2017]. It has grown into an open standard for 3D content on the web satifying the requirements of simplicity, compatibility, quality, interactivity and standardisation. As already described in section 4.3.1, we have used Three.js library, which is a high level API built upon WebGL.

There are a number of options for 3D rendering using WebGL. Babylon.js, PlayCanvas, Scene.js, A-Frame (VR), Three.js, etc. are just a few of these. There is even a tertiary level framework called Whitestorm.js which is built on top of three.js. We could even use WebGL as it is, without going for a higher level API. We see the pros and cons associated with each of them, but a benefit of using an open standard standard is that most of these secondary

Figure 5.3: Web3D rendering frameworks

frameworks are also open source. When we look at three.js, it is a kind of moderation between high and low level API – it is high-level enough to get rid of the issues related to browser and hardware interaction, while at the same time low-level enough to allow for flexibility within the code. If we look on the other hand, at Babylon.js, for example, it is at a much higher level with much logic already defined, and less flexibility in implementing your code. Then there are libraries like Paper.js which help in granular access by providing DOM-like representation of graphics, but they are primarily limited to 2D graphics. Many frameworks allow for importing and exporting functionalities, which was also required in our case. So, having loaders for Collada or glTF files within these frameworks didn't prove to be a major problem. All in all, three.js was best suited for providing 3D rendering environment with sufficient framework for interactivity, external files support and integration with other front-end frameworks like React.js. This choice is further cemented by a large and active community support, which is essential when trying to learn and use it for the first time.

5.2.3 Connecting Renderer to Graph Database

Once we have the three.js library, we need to look for a way to connect it to our semantic building data, which is stored in a Neo4j graph database. Neo4j has an active and supportive development community which has provided drivers for major programming languages including JavaScript. It provides us with an end point of an API through which we can run queries which can read, write and update data in the graph. However, there were two options on how to include it into our application, which have been defined earlier in section 4.4.3. Another choice was of the protocol used to connect - HTTP or binary. The driver uses binary protocol, whereas HTTP can directly be used with `request` node-module in JavaScript. We went for the binary one as it is officially supported and aims to be minimal, while allowing subscription to a stream of responses, errors, and completion events, as mentioned by the

developers.

5.2.4 Creating a Scene in Three.js

Once we have the identity of the elements required to be rendered from the graph database, we need to display them, within what is called a scene inside three.js. A scene is something which allows us to set up what is rendered and where. A description of three.js has already been presented in section 4.3.1.3. Here we try to focus more on its application.

If we look back at the scene graph in figure 4.11a, the highest element in hierarchy is the renderer, which is different from the context in which renderers have been talked about above. Three.js offers various renderers like *CSS3DRenderer*, *SVGRenderer*, etc. but the *WebGLRenderer* is the main highlight which uses WebGL technology and is the most versatile, with support from most of the modern browsers. Then, there are a number of cameras available in three.js, each with its own speciality. We use the Perspective camera, which serves a real world like view to the scene. It takes in certain properties like the field of view (FOV), aspect ratio, near and far clipping planes. Once we have these basic elements set up, we can then go on to define geometries and materials as a part of the mesh using the pre-defined constructs available within three.js (Figure 5.4). It also supports loading of external assets stored in various formats using the loaders defined for each of them. This is what we will be looking forward to in the next section.

```
 9   var geometry = new THREE.BoxGeometry( 1, 1, 1 );
10   var material = new THREE.MeshBasicMaterial( {
     color: 0x00ff00 } );
11   var cube = new THREE.Mesh( geometry, material );
12   scene.add( cube );
13
14   camera.position.z = 5;
```

Figure 5.4: Defining basic geometry within Three.js

5.2.5 Loading elements from External Files

We have our building elements available in the form of glTF assets, which can be loaded onto the three.js canvas using the *GLTFLoader* available as a part of the examples of three.js (Figure 5.5a). The usage in our case is slightly changed as we are using it inside a React.js environment, for the

client-side rendering, which uses webpack to bundle, compile, and transpile the code. In such a scenario, the loader is added as an `npm` package, which offers a better way to include the loader code into the application and eases its usability, as compared to directly importing the loader into the HTML. We can then use it as has been shown in figure 5.5b. The resulting scene on loading varying number and kind of elements of an office building model is shown in figure 5.6. Although initially this proved to be a challenge when working inside React.js framework, but was later resolved as explained in section 5.4.3. The files are named by the GUID number of the element they contain, and are thus easily traceable.

(a) Inside Three.js (b) Inside React.js

Figure 5.5: Example GLTFLoader Usage

5.2.6 Relative Positioning of Elements

Another possibility of a problem was that while loading the objects independently, their relative position, which was present in the complete model, might get lost. We had thought of solutions to it like loading each element on a separate canvas, and then arranging them, or even on the same canvas by positioning using three.js constructs. But for both COLLADA as well as glTF files, the elements retain their relative positions. This is due to the fact that the information about their positioning and the respective coordinate system remains unchanged when converting individual elements out of

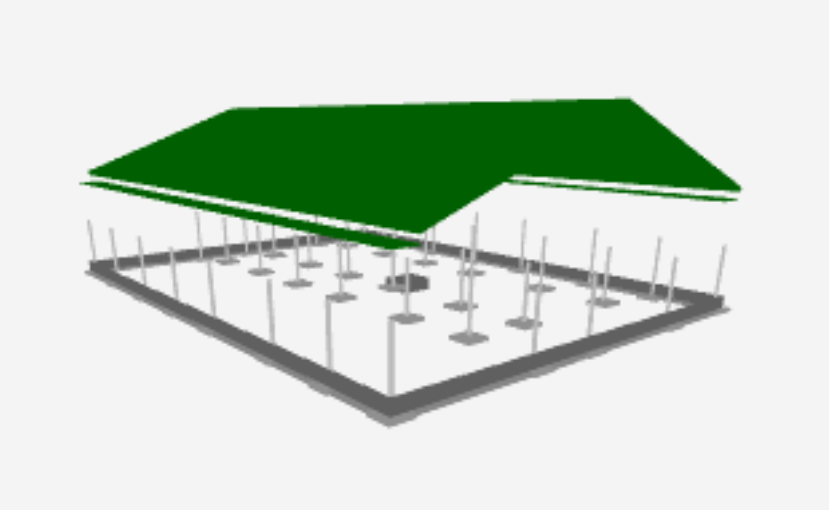


Figure 5.6: Partial Rendering of an Office Building Model

IFC. This can be attributed to the IfcConvert function which does the job originally, and then it is further preserved while converting to glTF as well.

5.3 Interactivity

For a better user experience in a 3D rendering environment, interactivity is really important. We identify two levels of interactivity which can be offered in our case – controlling what has already been served, and controlling what should be served. Separate approaches are required to address both of these, and separate utilities of three.js have been used to implement them. For the former, it is quite straight forward using the controls provided in the examples with three.js. There are two options – Orbit Controls (for third-person view) and Pointer Lock Controls (for first-person view). Orbit Controls better suite our requirement for building models, wherein we can rotate and zoom the model using mouse input.

5.3.1 On-Click Capture - Possible Options

The next and more complicated part was about controlling what should be served. This needs user to identify and select from already rendered objects. Thus, capturing elements on click inside a 3D environment becomes really important. We looked for possibilities of how this could be realised. The basic principle for this is to capture the mouse click position on the screen, and check for the element in that area of the screen. Building elements don't

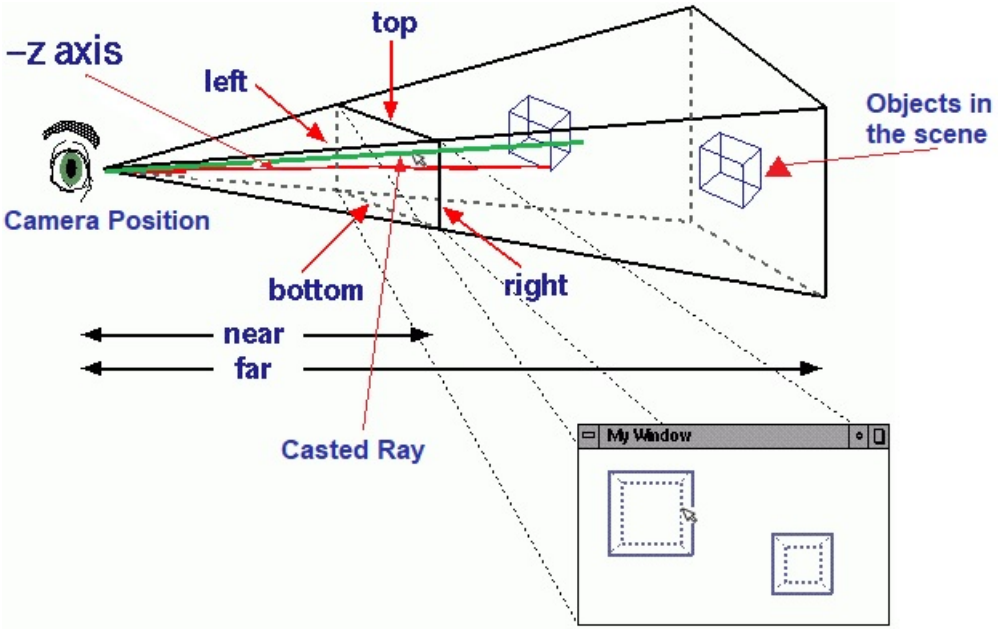


Figure 5.7: Illustration of 3D projection on 2D screen (Recreated from source [Game Development])

generally have uniform shapes. Especially when rendering 3D models, different elements are displayed at different angles of projection and combining this with their complex shapes, it is really challenging to exactly outline the elements on the screen. Thus, a necessary and useful approximation is made by considering a tight rectangular box around the element. This concept of bounding box (BB) has been extensively used in image annotations, and even in AEC domain for applications like clash detection [Helm et al., 2010].

Thus, we find that IFC also provides its own *IfcBoundingBox* entity as a part of *IfcGeometricRepresentationItem*. But the problem with IFC representation is that it uses relative positioning for elements thus giving each one a different coordinate system. A necessary thing for mapping these BBs with the screen click position is to have a common coordinate system. Although all the elements can be traced back to one root coordinate system in the IFC, it requires computations. A better way is to have a single reference system while generating the BBs itself. Another method for the same using the Python implementation of Open Cascade, which is an open source library for 3D modeling and visualisation, has been used by Zibion [2018]. There the use of bounding boxes was primarily for inter element transactions like clash detection, etc. For comparing with the screen coordinates, we needed to get into the Three.js renderer and how it actually projects the scene onto

the screen.

5.3.2 Three.js Bounding Box

Three.js library is meant for all kinds of interactive 3D visualisation. Thus, it also provides a bounding box concept of its own for the elements which are loaded onto the scene. Benefit with this is that all elements are represented in a single reference system on the canvas. Further, we have the details of camera and projection, which connects the scene information to the plane of the display screen, thus connecting it to mouse events. But since the scene is in 3D, overlaps between objects are possible, thus, we need to use the depth of the elements relative to display screen in order to identify them correctly. In order to account for all this, we use a method of casting a ray and then checking for its intersection with any objects, as explained below.

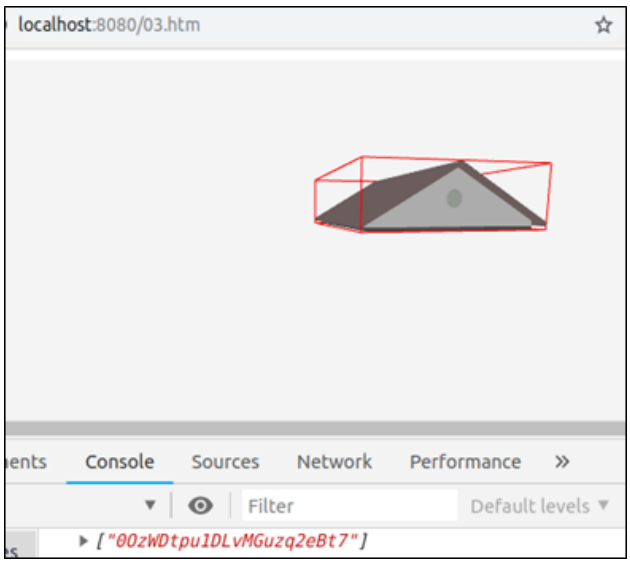


Figure 5.8: Building Storey with its bounding box in Three.js (Captured GUID can be seen in the console)

5.3.3 Intersection Using Raycaster

Three.js provides a feature for creating a ray between any two points using the Raycaster class. It is mainly used for mouse picking in a 3D environment, as mentioned in the official documentation. We use a predefined method of `raycaster.setFromCamera(mouse, camera)`, which generates a ray from the position of the camera to the point of the mouse. The mouse position

needs to be converted into normalised device coordinates (NDC) (between -1 and 1) for both x and y direction, using the following equations:

$$x_{norm} = (x_{abs}/w_{can}) * 2 - 1; \tag{5.1}$$
$$y_{norm} = -(y_{abs}/h_{can}) * 2 - 1; \tag{5.2}$$

where:

x_{norm}, y_{norm}	Mouse coordinates in NDC
x_{abs}, y_{abs}	Absolute mouse coordinates w.r.t client area
w_{can}, h_{can}	Width and height of the client area respectively

These normalised coordinates are then used for generating/updating the raycaster. Once we generate the ray, the `raycaster.intersectObjects (object, recursive)` function is used to check the intersection with objects passed in the '`object`' field. The '`recursive`' field takes a boolean value – `true` to consider the descendent elements of the objects, and `false` otherwise, which is also the default case. This returns a sorted list of all intersecting elements, with closest first.

```
var mouse = new THREE.Vector2();

function onMouseDown(event) {
    event.preventDefault();

    mouse.x = (event.clientX / renderer.domElement.clientWidth)* 2 - 1;
    mouse.y =  - (event.clientY / renderer.domElement.clientHeight)* 2 + 1;
```

(a) Code implementation of equations 5.1 and 5.2

```
raycaster.setFromCamera(mouse, camera);
var intersects = raycaster.intersectObjects(objects, true);
if (intersects.length>0) {
  // GUID can be accessed by intersects[0].object.object.children[0].name
  // Query Graph DB with the GUID number
}
```

(b) Applying intersection check within the 'render' method

Figure 5.9: On-click element capture using Raycaster

5.3.4 Challenges Faced

A few difficulties were faced while implementing this solution. The object capture was not working accurately during initial tries. Some points within the BB were not being captured, while some other just outside the BB were being recognised as a part of it. We must remember that the user has controls like moving and zooming in and out the scene. The problem being encountered was that after a change in orientation using user controls, the casted ray was not being updated each time, which led to inaccuracies. The problem was identified and rectified within the code to re-create the ray every time controls are updated. Thereafter fresh intersection checks are done, leading to correct results.

5.4 Integration with 2D platform

Although we have created a stand-alone prototype based on our concept, but still having a seamless 2D + 3D rendering is a treat on the top. We have already talked about how both 2D and 3D have their own benefits and drawbacks, and an option for both would increase the richness of our model, filling the gaps left by each one.

5.4.1 Characteristics of 2D Platform

The 2D rendering [Zibion, 2018] is done using floor plans which are converted from IFC to SVG format. The SVG is an XML-based format, and it becomes a part of the HTML DOM tree when rendering using React.js front-end framework. This makes it possible to avail and load only parts of the plan as and when they are required using simple DOM transactions. Thus, we get a granular access to geometrical elements.

5.4.2 Possible Options for Integration

There can be two ways for integration – 3D rendering using the platform used for 2D or the other way round. Whenever we load a 3D model onto a web page, it renders using the HTML5 canvas element. A drawback of canvas is that it hoods the entire scene within itself, and there is no access to the individual scene elements while online even for XML file formats. This goes against the concept on which SVG floor plans were implemented. So, if we try to put 2D files inside a 3D canvas environment, it would loose its granular capability. Thus, we go for the other way, i.e., placing 3D rendering

within the environment used for 2D. This is one of the major reasons for using React based rendering along with Three.js.

5.4.3 Three.js inside React.js

React.js has its own environment which runs using webpack for code bundling and compilation. Using raw Three.js does not require any such steps. Thus, if we try to put raw Three.js code directly inside React, it doesn't work. We, therefore, import Three.js and other related dependencies like GLTFLoader, OrbitControls as node modules through npm. Initial trials were done by defining simple geometry within Three.js and worked fine. But while loading external files, there was a problem as the files could not be loaded in a webpack environment. The problem identified was that webpack does not resolve relative file paths, and so the loader couldn't reach the file in the first place, let alone render it. In order to solve this issue, we resolved and imported the path to file separately inside the React component, and then used this already resolved path inside the glTF loader function, which then rendered it correctly. Thus, we were able to successfully load partial models using Three.js inside a React framework and this can now be combined with React-based 2D renderer easily.

Chapter 6

Discussion

We have seen until now how the knowledge from various domains is related to the problem we are addressing, and how it has been tackled in the current work. Now, some of the important aspects of the evaluation and discussion, as also mentioned previously, will be taken up in this chapter, followed with some of the limitations of the research.

6.1 In Context of AEC/FM Industry

We are primarily working within the BIM paradigm, which in-turn is a part of the AEC/FM industry. Thus, it is important to see how the current work relates to the industry and how it can actually be utilised in that context. Thus, we will discuss it in the perspective of three possible use cases spanning different stages of the building lifecycle, as follows.

6.1.1 Facility Management

Facility Management (FM) is a recent and growing domain which deals with integration and collaboration of people and processes from specialised fields in order to enhance the process of operations and maintenance in a built facility. This idea of FM has been supported by the International Facility Management Association (IFMA) as well as some independent researchers like Atkin and Brooks [2015]. Consequently, FM receives an influx of large information from various fields (Figure 6.1), which causes the same problems related to IR.

Becerik-Gerber et al. [2011] points out the different application areas in FM where BIM can be implemented, by conducting a survey among the users and non-users of BIM (Figure 6.2). They have further laid out the challenges

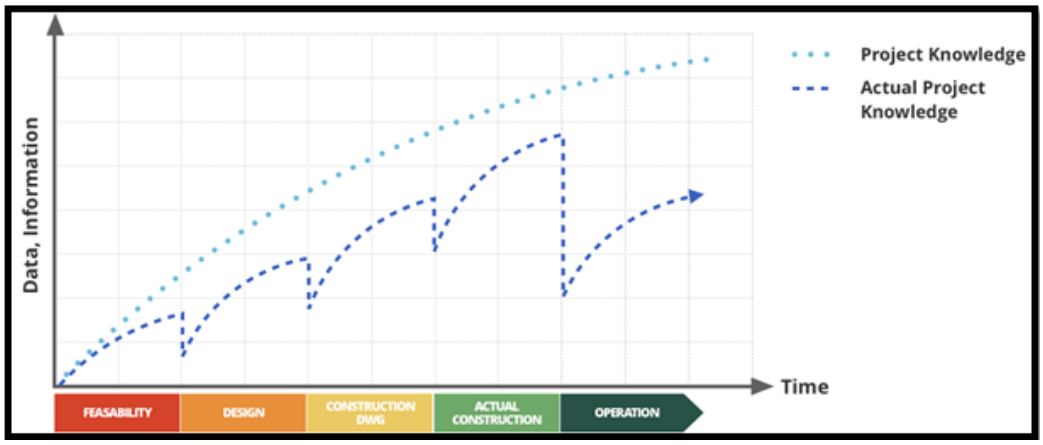


Figure 6.1: Information in FM [Zibion, 2018]

in implementation of BIM in FM, dividing them into two categories relating to organisational and technical aspects. Despite some challenges, there is no denying the benefits of BIM in FM, as have been documented by Codinhoto and Kiviniemi [2014] in terms of significant reductions in man hours and waiting time per FM task.

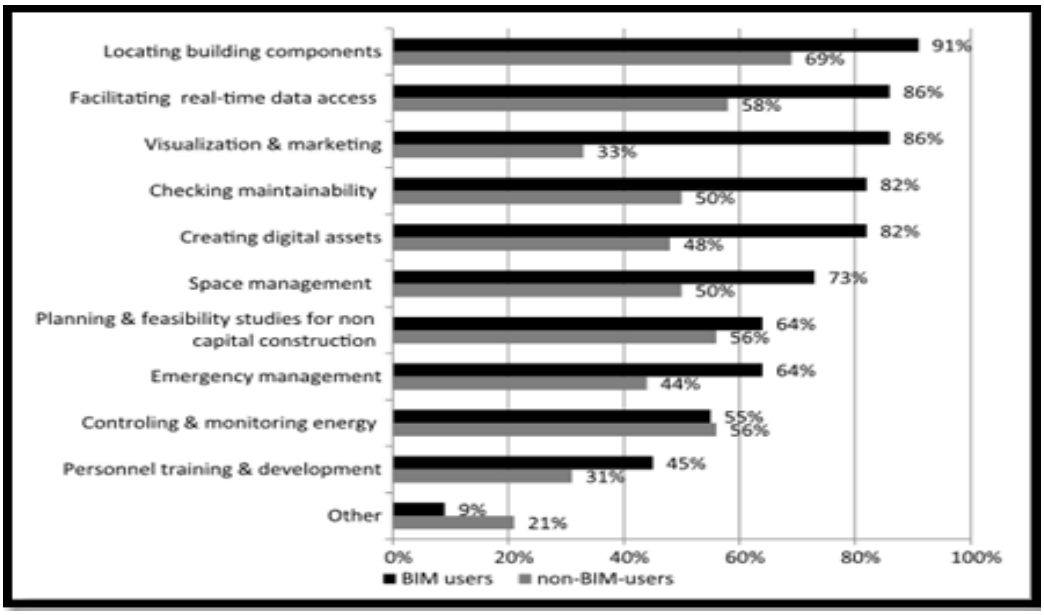


Figure 6.2: Potential BIM in FM Application Areas [Becerik-Gerber et al., 2011]

Although mainstream CMMS tools don't use any kind of 3D or 2D models

currently because of a general lack of knowledge of CAD-type tools among the FM personnel, but the opportunities of data use and interoperability offered by BIM can not be ignored. But this has to be done in such a way which does not create difficulties in the tasks of FM practitioners. This can only be achieved by simplifying the complex BIM models, and making their use more intuitive and independent of knowledge of CAD. This is true for both 2D and 3D models. We argue that a combination of simplified 2D and 3D models working at different scales of granularity can provide a comprehensive solution to the problems of FM practitioners in BIM.

6.1.1.1 An FM Use Case Scenario

Let us look at a simple flow of work order as shown in figure 6.3. The occupants inform the facility manager about the room number and the system which has problem (let's say that an electric light is not working). The manager then identifies the room on the 2D floor plan of the respective level and the addressed faulty system (Electrical in this case). He then opens a partial 3D model of that single room (from the MEP and architectural models), wherein he identifies the problematic part and refers to the technician. The technician takes a closer look at the reference by the manager and identifies the faulty light and the tools required to fix it.

Figure 6.3: Simplified FM Workflow Example (Adopted from Zibion [2018])

He sees from the log of tasks that this light was replaced just a few days ago, and shouldn't have gone out so early. So, he takes a look at the electrical network of that room through the partial 3D model. From within the bunch of intermingled wire network, he can highlight the required connection from

the light to its switch because of the semantic information relating these elements. He then finds his way to the location using the 2D plan and once he reaches the room, he can again open the partial 3D model on his mobile device using just a web browser.

After physically inspecting the nature of problem, he gets to know of some water leakage near the port, so he loads only the related elements from the plumbing model on top of the partial electric model and identifies a plumbing joint near the connecting electrical line which was repaired a day before. He now knows the position behind the walls where the water would have caused the wires to short circuit, and can thus, address the real problem efficiently.

6.1.1.2 Remarks

The presence of lightweight partial 3D models facilitates the assessment of problems in operations on-the-go, leading to shorter down-times and faster maintenance, ultimately leading to a better quality of service to the users and ease in job for the facility managers. In addition, using BIM data saves the otherwise heavy efforts for data transfer from construction to operations, and the subsequent log management, which are now directly connected and stored with the model itself.

6.1.2 Construction Progress Monitoring

Moving our attention back to the construction process, we see that the productivity of the construction process has been discussed often. Progress monitoring is critical to increasing production rates. It allows the project managers to act in time to avoid time and cost overruns in cases of deviation from the schedule. Kopsida et al. [2015] have categorised this process into four steps involving data acquisition, information retrieval, progress estimation, and visualisation of results. We only intend to focus on the estimation part, which consists of matching the information captured from the as-built model to the as-designed model. The data is in the form of point clouds or images and videos. In order for proper comparison, the designed model needs to be used partially according to the stage of construction. Further, even the designed model evolves during the early stages. Thus, we can even create a versioning system of the designed model, with new elements being added in successive versions and previous ones getting modified. These versioned elements can better convey the design changes to the personnel on site, and also directly be used for inspection and monitoring purposes.

A lot of focus in research has been towards the automation of the progress monitoring process, wherein the data is acquired, processed and compared at

fixed intervals automatically, and can be viewed on demand. Generally, the as-built model is superimposed on top of the as-designed model, a process known as registration. After this, element wise comparison can be made. Various methods have been deployed for this purpose, based on the type of collected as-built data including 2D images, point clouds, etc. Many of these try to compare it to 4D BIM models (with 4^{th} dimension being time) using methods like reconstructing point clouds from images [Golparvar-Fard et al., 2012], surface matching [Turkan et al., 2012], counting number of points inside point clouds [Zhang and Arditi, 2013], etc. Some research has also considered using methods involving crane camera images and comparing floor-wise to the as-designed models. We propose that having granular as-designed models in combination with the work schedule determining which elements must be present at a particular time, can be used for comparisons with as-built models. Different elements can be tagged with scheduled dates for their completion inside the BIM model, and this can then be used to extract a group of all the elements supposed to be completed by a given date at which we capture the as-built model, and then compare them.

This is an area of possible application of the proposed method of granularity of building models, and is open for further specialised development using the basic concept presented in this thesis.

6.1.3 Construction Site Situational Awareness

The temporary loss or lack of situational awareness is a causal factor in many construction accidents [HSE, 2014]. Construction site situational awareness involves assessing and utilising the knowledge of how things are working on the ground. This awareness can be of the workers, supervisors, or the managers. Also, this information needs to be temporally contextual. But not all situations are needed to be conveyed to everyone. Suppose a worker is assigned for a plumbing job for a particular section of a particular floor on a given day. He might not necessarily require the details related to a carpenter working in an entirely different part of the building. So, the information also needs to be spatially contextual. Thus, granularity is desired in the building models which are conveyed to him. A web-based granular platform can help provide only the necessary part of the design to the worker on site on any given day, and it might also include his own schedule to help him better plan the day's work.

The identification of the positions of workers on site can be done using various tracking technologies like attaching beacons, etc. The located space can be connected to the corresponding spatial element in the IFC (like IfcSpace for rooms, etc.) and building elements within that space can be

granularly extracted using the proposed approach and fed to their mobile devices.

6.2 In Context of Information Retrieval

The concept of IR has proved to be a theoretical backbone of this work. But how does the proposed work fare in terms of IR needs to be seen. We try to improve the IR process by incorporating the relevance, speed and accuracy to the visualisation process. By providing tailored models for every scenario, we improve the relevance of the information being served. It not just retrieves the matching elements, but rather looks at their relations and helps serve the larger purpose for which they are being retrieved. This contextualisation has been two-fold – one is the interrelation between different elements and extracting other elements based on these connections, the other is the information about the elements like material properties, cost requirements, ownership details, maintenance logs, etc. which is present inside the IFC and can be successfully translated to graph data structure. Further, by having partial models instead of entire, the size is significantly reduced which should normally lead to an improvement in speeds, although it still needs to be tested. Thus, we try to overcome the problems faced by current visualisation platforms.

Another place where IR plays a special role is in the use cases mentioned above. Both progress monitoring and situational awareness require information retrieval in their own form. For example in progress monitoring, data from built structure is captured and relevant information for comparison needs to be retrieved from it. In fact, Kopsida et al. [2015] mention IR as one of the four steps of the progress monitoring process. Techniques like photogrammetry, image processing, and computer vision have been applied to extract 3D models from captured images or point clouds – all of these involving IR in some capacity. Same is the case of situational awareness, where contextual relevance of information is necessary to improve the performance on ground. Thus, we see that incorporating granularity in BIM models helps improve the process of IR.

6.3 Limitations of the Research

As we have mentioned before and would like to re-iterate over here, no research effort is perfect in itself and there ought to be new challenges and weaknesses which are discovered as we go along. Same is the case for us.

While we argue that granularity in BIM is important and provide our evidences for that, it is important to note that the same level of granularity cannot be efficient for all purposes. Thus, although we try to increase the level of granularity to the element level, some applications might actually not be in need of it and would prefer working with entire models. Applying the same concept of granular loading to entire models might actually break down the application due to memory limits. The combined size of all the partial files was observed to be greater than that of the single IFC file of the entire model. This might have its own reasons like redundancy of certain information which is transferred to each of the files in order to maintain their cohesiveness, and so on. But the point is that in situations like preparing project budget estimations, for example, where whole model is needed to be utilised at the same time, it won't be a good idea to go granular to the element level.

Further, this work is only an experimental and localised implementation of the proposed concept serving as a proof of its feasibility. The user interface and overall aesthetics of the application can be improved in a full scale deployment of this concept to make it more intuitive and user friendly. Also, we have restricted to descriptive methods of evaluation based on the characteristics of the problems and the solution, and the requirements it tries to fulfill. Detailed analytical testing and evaluation, like that for speed, etc. was out of scope of the current work and has been left out for future. A feedback from the professionals in the industry and how they think about the element-level granular approach might also provide useful insights and is something which is missing from the current work. But despite these challenges, we feel that the idea of granularity which we wanted to convey has been tackled sufficiently well, and the conceptual and methodological contributions from this work can be considered something to be looked at.

Chapter 7

Conclusions

Until now, we have discussed about BIM, its role in the AEC industry, and how granularity is important in this context. We've also seen how the industry is gradually moving towards granularity with the concepts like linked building data and semantic web, and how current visualisation platforms are lagging on this front. We have then laid down the requirements for a BIM-based visualisation platform, and introduced a new concept to incorporate granular approach within the visualisation of 3D BIM models. We also show how having both 3D and 2D partial models integrated on a single platform can provide a more wholesome solution. In order to support this concept and prove its feasibility, we have then implemented a prototype solution which is based upon open standards. We have highlighted the various challenges faced during the implementation and how they were tackled. Later, we have discussed the opportunities and possible applications in the industry for the presented concept by providing example scenarios and how this approach can help in such cases. The contribution towards the knowledge domains like information retrieval has also been talked about briefly. Finally, some of the limitations of the current work have been highlighted.

We conclude over here, looking back at the research questions we had mentioned in the beginning and how they have been answered. The first question dealt with the possibilities of 3D BIM models being modularised for visualisation. We see that by having an IFC file split into individual geometric components, we can indeed access and visualise them independently, thus, providing a way for granularity. This concept has been successfully implemented in the form of a web-based application. We see that the gap between the modular building information (in the form of graph based data structures) can be successfully connected on the same level of granularity to a visualisation solution, with both being based on the web.

The second question concerned more about the application of such an ap-

proach within the industrial scenarios. We have explained how in the case of Facility Management, the granular approach helps improve the performance in the maintenance tasks, by providing targeted and quick analysis of the situations. Performance within the construction process can also be analysed and eventually improved through tasks like progress monitoring and site situation management. The workers can remain focussed on their work and perform well if they are briefed with the proper requirements of the design through simple visualisations of only the sections relevant to them on a daily basis. They can even refer to that on-the-go using their mobile devices. Thus, we see that the presented approach can really help in many scenarios during the construction process and beyond. We now move on to provide an outlook into the future prospects and improvements related to the current work.

7.1 Further Scope

BIM has really grown into a vast area of interest, both from the point of view of the industry and academia. We see the industry moving from the open IFC standard of building models towards the RDF graph-based data representation in view of the semantic web. Approaches like SimpleBIM, etc. have emerged and we follow a similar approach using a different kind of graph database, Neo4j. RDF-graph based IfcOWL and other advancements to it can be used in place of Neo4j graphs, and can enable connections of IFC geometry to semantics over the semantic web. Further, the current prototype can be developed into a full scale, possibly cloud-based implementation, which can then be used for pilot testing within the industry. As a part of the analysis and testing, interviews with the industry professionals can be conducted to get their feedback, thus continuing the cycle of design iterations.

Apart from this, features from GIS can be incorporated on a city level, for example, to provide seamless indoor plus outdoor visualisation. Another important development in our view from the point of view of granularity is the back-end architecture. The current system stores the entire graph database and the related partial models on a single server. What we plan to implement is a 'Server Less' architecture where the graph would be modularised into multiple sub-graphs and stored at separate locations connected over the web. The processing load along with the data would thus be distributed, providing another aspect to the granularity of BIM. Further, many other application domains outside construction and FM can be thought of for applying similar principles of granularity. And better management of the vast amounts of

data, most of which is being collected without proper aim and use, can be achieved.

Bibliography

Brian Atkin and Adrian Brooks. *Total facility management*. John Wiley & Sons, 2015.

Salman Azhar. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadership and management in engineering*, 11(3):241–252, 2011.

Salman Azhar, Abid Nadeem, Johnny YN Mok, and Brian HY Leung. Building Information Modeling (BIM): A new paradigm for visual interactive modeling and simulation for construction projects. In *Proc., First International Conference on Construction in Developing Countries*, volume 1, pages 435–46, 2008.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

Raphael Barbau, Sylvere Krima, Sudarsan Rachuri, Anantha Narayanan, Xenia Fiorentini, Sebti Foufou, and Ram D Sriram. OntoSTEP: Enriching product model data using ontologies. *Computer-Aided Design*, 44(6):575–590, 2012.

Mark Barnes and Ellen Levy Finch. Collada–digital asset schema release 1.5. 0 specification. *Khronos Group, Sony Computer Entertainment Inc*, 2008.

Florian Bauer and Martin Kaltenböck. Linked open data: The essentials. *Edition mono/monochrom, Vienna*, 710, 2011.

Burcin Becerik-Gerber, Farrokh Jazizadeh, Nan Li, and Gulben Calis. Application areas and data requirements for BIM-enabled facilities management. *Journal of construction engineering and management*, 138(3):431–442, 2011.

Jakob Beetz, Jos Van Leeuwen, and Bauke De Vries. Ifcowl: A case of transforming express schemas into ontologies. *Ai Edam*, 23(1):89–101, 2009.

Nicolas J Belkin and WB Croft. Information retrieval and information filtering: two sides of the same coin. *Communications of the ACM*, 35(12): 29–38, 1992.

Tim Berners-Lee and Mark Fischetti. Weaving the Web. HarperSanFrancisco. Chapter 12. 1999.

André Borrmann, Markus König, Christian Koch, and Jakob Beetz. *Building Information Modeling: Technologische Grundlagen und industrielle Praxis*. Springer-Verlag, 2015.

BuildingSMART-Tech. Ifc guid. `http://www.buildingsmart-tech.org/implementation/get-started/ifc-guid`. (Accessed on 05/20/2019).

BuildingSMART-Tech. Ifc introduction. `http://www.buildingsmart-tech.org/ifc/`. (Accessed on 05/20/2019).

BuildingSMART-Tech. Structural curve member. `http://www.buildingsmart-tech.org/ifc/examples/structural-analysis-model/structural-curve-member.htm`, 2019a. (Accessed on 05/11/2019).

BuildingSMART-Tech. Column with straight extrusion. `http://www.buildingsmart-tech.org/ifc/examples/building-element-standard-case/column-extruded-solid.htm`, 2019b. (Accessed on 05/11/2019).

Client–Server Model. Client–server model - wikipedia. `https://en.wikipedia.org/wiki/Client%E2%80%93server_model`, 2019. (Accessed on 05/21/2019).

Ricardo Codinhoto and Arto Kiviniemi. Bim for fm: a case support for business life cycle. In *IFIP International Conference on Product Lifecycle Management*, pages 63–74. Springer, 2014.

Ricardo Codinhoto, Arto Kiviniemi, Sergio Kemmer, and Cecilia Gravina da Rocha. BIM-FM implementation: an exploratory invesigation. *International Journal of 3-D Information Modeling (IJ3DIM)*, 2(2):1–15, 2013.

William S Cooper. A definition of relevance for information retrieval. *Information storage and retrieval*, 7(1):19–37, 1971.

CRC Construction Innovation. Adopting BIM for facilities management: Solutions for managing the Sydney Opera House. *Cooperative Research Center for Construction Innovation, Brisbane, Australia*, 2007.

Ray Crotty. *The impact of building information modelling: transforming construction*. Routledge, 2013.

Carlos A Cuadra. *Experimental Studies of Relevance Judgments. Final Report*. System Development Corporation, 1967.

NN Dawood and Nahim Iqbal. Building information modelling (BIM): A visual & whole life cycle approach. pages 7–14. CONVR2010 Organizing Committee, 2010. ISBN 9784990553708C3000.

DB-Engines. DB-Engines Ranking - popularity ranking of graph DBMS. `https://db-engines.com/en/ranking/graph+dbms`, 2019. (Accessed on 05/20/2019).

Scott C Deerwester, Susan T Dumais, George W Furnas, Richard A Harshman, Thomas K Landauer, Karen E Lochbaum, and Lynn A Streeter. Computer information retrieval using latent semantic structure, June 13 1989. US Patent 4,839,853.

Bing Dong, Zheng O'Neill, and Zhengwei Li. A BIM-enabled information infrastructure for building energy Fault Detection and Diagnostics. *Automation in Construction*, 44:197–211, 2014.

Diogo Fernandes and Jorge Bernardino. Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In *Proceedings of the 7th International Conference on Data Science, Technology and Applications,{DATA}*, pages 373–380, 2018.

Neil D Fleming. *Teaching and learning styles: VARK strategies*. IGI global, 2001.

Game Development. Graphics - How do 3D game engines render 3D environments to a 2D screen? `https://gamedev.stackexchange.com/questions/10030/how-do-3d-game-engines-render-3d-environments-to-a-2d-screen`. (Accessed on 05/27/2019).

Ali Ghaffarianhoseini, John Tookey, Amirhosein Ghaffarianhoseini, Nicola Naismith, Salman Azhar, Olia Efimova, and Kaamran Raahemifar. Building Information Modelling (BIM) uptake: Clear benefits, understanding its implementation, risks and challenges. *Renewable and Sustainable Energy Reviews*, 75:1046–1053, 2017.

GitHub - Three.js. Github - mrdoob/three.js: Javascript 3d library. `https://github.com/mrdoob/three.js`. (Accessed on 04/24/2019).

GLTF 2.0 Specifications. gltf/specification/2.0 at master · khronosgroup/gltf · github. `https://github.com/KhronosGroup/glTF/tree/master/specification/2.0`. (Accessed on 05/20/2019).

Mani Golparvar-Fard, Feniosky Peña-Mora, and Silvio Savarese. Automated progress monitoring using unordered daily construction photographs and ifc-based building information models. *Journal of Computing in Civil Engineering*, 29(1):04014025, 2012.

Pim van den Helm, Michel Böhms, and Léon van Berlo. IFC-based clash detection for the open-source BIMserver. In *Computing in civil and building engineering, proceedings of the international conference. Nottingham University Press, Nottingham, UK*, volume 181, 2010.

Lars Madsen Hestman. The Potential of Utilizing BIM Models With the WebGL Technology for Building Virtual Environments-A Web-Based Prototype Within the Virtual Hospital Field. Master's thesis, NTNU, 2015.

Jiri Hietanen and Sakari Lehtinen. The useful minimum. Technical report, Working paper. Tampere University of Technology. Virtual Building Laboratory, 2006.

Nam Vu Hoang and Seppo Törmä. Implementation and Experiments with an IFC-to-Linked Data Converter. In *Proceedings of the 32nd International Conference of CIB W78*, pages 285–294, 2015.

JE Holmstrom. Section III. Opening plenary session. In *The Royal Society Scientific Information Conference, 21 June–2 July 1948: Report and papers submitted*, 1948.

HSE. Situational awareness. `http://www.hse.gov.uk/construction/lwit/assets/downloads/situational-awareness.pdf`, September 2014.

Information Retrieval Lab - CSUI. Information retrieval. `http://ir.cs.ui.ac.id/new/`, 2016. (Accessed on 05/11/2019).

Ali Ismail, Ahmed Nahar, and Raimar Scherer. Application of graph databases and graph theory concepts for advanced analysing of BIM models based on IFC standard. *Proceedings of EGICE*, 2017.

Dean Jackson. Next-generation 3D Graphics on the Web — WebKit. `https://webkit.org/blog/7380/next-generation-3d-graphics-on-the-web/`, February 2017. (Accessed on 05/10/2019).

A Khalili and DK H Chua. IFC-based graph data model for topological queries on building elements. *Journal of Computing in Civil Engineering*, 29(3):04014046, 2013.

Marianna Kopsida, Ioannis Brilakis, and Patricio Antonio Vela. A review of automated construction progress monitoring and inspection methods. In *Proc. of the 32nd CIB W78 Conference 2015*, pages 421–431, 2015.

Sylvere Krima, Raphael Barbau, Xenia Fiorentini, Rachuri Sudarsan, and Ram D Sriram. Ontostep: OWL-DL ontology for step. *National Institute of Standards and Technology, NISTIR*, 7561, 2009.

Mikael Laakso, AO Kiviniemi, et al. The IFC standard: A review of history, development, and standardization, information technology. *ITcon*, 17(9): 134–161, 2012.

Josep Lluís Larriba-Pey, Norbert Martínez-Bazán, and David Domínguez-Sal. Introduction to graph databases. In *Reasoning Web International Summer School*, pages 171–194. Springer, 2014.

Thomas Liebich. Unveiling ifc2x4-the next generation of openbim. In *Proceedings of the 2010 CIB W78 Conference*, volume 8, 2010.

Xiaojun Liu, Ning Xie, Kai Tang, and Jinyuan Jia. Lightweighting for Web3D visualization of large-scale BIM scenes in real-time. *Graphical Models*, 88: 40–56, 2016.

David Scott Lyons. Intro to WebGL with Three.js. In *Front Porch Conference, Dallas, Texas*, 2014. (Accessed on 05/12/2019).

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1): 100–103, 2010.

Bernard Marr. How much data do we create every day? The mind-blowing stats everyone should read. *Forbes, May 21st*, 2018. (Accessed on 02/20/2019).

Tarcisio de Farias Mendes, Ana Roxin, and Christophe Nicolle. IfcWoD, semantically adapting IFC model relations into OWL properties. *arXiv preprint arXiv:1511.03897*, 2015.

Jeffrey Morgan. Requirements for Visualization Software. `https://usabilityetc.com/articles/visualisation-software-requirements/`. (Accessed on 05/13/2019).

Ali Motamedi, Amin Hammad, and Yoosef Asen. Knowledge-assisted BIM-based visual analytics for failure root cause detection in facilities management. *Automation in construction*, 43:73–83, 2014.

Ernesto Olivares. We are 90% visual beings. `https://ernestoolivares.com/we-are-90-visuals-beings/`, January 2013. (Accessed on 05/12/2019).

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Pieter Pauwels and Anna Roxin. SimpleBIM: From full ifcOWL graphs to simplified building graphs. In *Proceedings of the 11th European Conference on Product and Process Modelling (ECPPM)*, pages 11–18, 2016.

Pieter Pauwels and Walter Terkaj. EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology. *Automation in Construction*, 63:100–133, 2016.

Pieter Pauwels, Ronald De Meyer, and Jan Van Campenhout. Visualisation of semantic architectural information within a game engine environment. In *10th International conference on Construction Applications of Virtual Reality (CONVR 2010)*, pages 219–228, 2010.

Pieter Pauwels, Davy Van Deursen, Jos De Roo, Tim Van Ackere, Ronald De Meyer, Rik Van de Walle, and Jan Van Campenhout. Three-dimensional information exchange over the semantic web for the domain of architecture, engineering, and construction. *Ai Edam*, 25(4):317–332, 2011.

Pieter Pauwels, Thomas Krijnen, Walter Terkaj, and Jakob Beetz. Enhancing the ifcOWL ontology with an alternative representation for geometric data. *Automation in Construction*, 80:77–94, 2017.

Mads Holten Rasmussen, Pieter Pauwels, Christian Anker Hviid, and Jan Karlshøj. Proposing a central AEC ontology that allows for domain specific extensions. In *2017 Lean and Computing in Construction Congress*, 2017.

Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases: new opportunities for connected data*. " O'Reilly Media, Inc.", 2015.

Gerard Salton, Edward A Fox, and Harry Wu. Extended boolean information retrieval. Technical report, Cornell University, 1982.

Richard Saxon. Getting the dimensions of BIM into focus. `http://www.bimplus.co.uk/people/getting-dimensions-bim-focus/`, July 2018. (Accessed on 05/16/2019).

Hans Schevers and Robin Drogemuller. Converting the industry foundation classes to the web ontology language. In *2005 First International Conference on Semantics, Knowledge and Grid*, pages 73–73. IEEE, 2005.

Eric Schurman and Jake Brutlag. Performance related changes and their user impact. In *velocity web performance and operations conference*, 2009.

Davood Shojaei, Abbas Rajabifard, Mohsen Kalantari, Ian D Bishop, and Ali Aien. Design and development of a web-based 3d cadastral visualisation prototype. *International Journal of Digital Earth*, 8(7):538–557, 2015.

A Singhal and M Cutts. Using site speed in web search ranking. Google Webmaster Central Blog, 2010.

Peter Smith. BIM implementation–global strategies. *Procedia Engineering*, 85:482–492, 2014.

Eike Tauscher, Hans-Joachim Bargstädt, and Kay Smarsly. Generic BIM queries based on the IFC object model using graph theory. In *The 16th International Conference on Computing in Civil and Building Engineering, Osaka, Japan*, 2016.

Jaime Teevan, Kevyn Collins-Thompson, Ryen W White, Susan T Dumais, and Yubin Kim. Slow search: Information retrieval without time constraints. In *Proceedings of the Symposium on Human-Computer Interaction and Information Retrieval*, page 1. ACM, 2013.

S Törmä. Web of building data—integrating IFC with the web of data. In *Proceedings o f the 10th European Conference on Product and Process Modelling—eWork and eBusiness in Architecture, Engineering and Construction, Vienna, Austria*, page 141, 2014.

Seppo Törmä. Semantic linking of building information models. In *2013 IEEE Seventh International Conference on Semantic Computing*, pages 412–419. IEEE, 2013.

Yelda Turkan, Frederic Bosche, Carl T Haas, and Ralph Haas. Automated progress tracking using 4d schedule and 3d sensing technologies. *Automation in construction*, 22:414–421, 2012.

World Wide Web Consortium W3C. W3C Semantic Web Activity Homepage. `https://www.w3.org/2001/sw/`, 2011. (Accessed on 05/07/2019).

W3Schools. Xml tree. `https://www.w3schools.com/xml/xml_tree.asp`. (Accessed on 05/20/2019).

Chengyi Zhang and David Arditi. Automated progress control using laser scanning technology. *Automation in construction*, 36:108–116, 2013.

Jiansong Zhang. Towards Systematic Understanding of Geometric Representations in BIM Standard: An Empirical Data-Driven Approach. In *Construction Research Congress 2018: Construction Information Technology*, 2018.

Le Zhang and RR Issa. Development of IFC-based construction industry ontology for information retrieval from IFC models. In *Proceedings of the 2011 Eg-Ice Workshop, University of Twente, The Netherlands*, volume 68, 2011.

Daniel Zibion. Development of a BIM-enabled Software Tool for Facility Management using Interactive Floor Plans, Graph-based Data Management and Granular Information Retrieval. Master's thesis, TUM, 2018.