Technical University of Munich

TUM School of Engineering and Design

Chair of Computational Modelling and Simulation

Opportunities and current limitations of cloud-based design automation in the context of Building Information Modelling

Master Thesis

for the Master of Science program in Civil Engineering

Author: Jovin John

Matriculation Number: [redacted]

1. Supervisor: Prof. Dr.-Ing. André Borrmann

2. Supervisor: Sebastian Esser M.Sc.

Jimmy Abualdenien M.Sc.

Date of Issue: 15. June 2021

Date of Submission: 29. December 2021

Abstract

Integration of Cloud Computing in Building Information Modelling (cloud-BIM) can be considered as the second generation of BIM development. The benefits of cloud computing can solve many limitations faced by the conventional software tools in BIM, such as the requirement of high-end hardware, low accessibility and scalability, high license fees for on-premise software, etc. The possibility to update the BIM models online without the use of BIM authoring tools could help various stakeholders and make BIM more transparent. Apart from employing cloud computing in Common Data Environment (CDE), the high-performance computing capabilities can also be utilised for outsourcing tedious and repetitive design tasks in daily BIM workflows. The human capacity to complete laborious, repetitive design tasks in a specific timeframe is limited. Automating such time-consuming design tasks helps various stakeholders save time for more creative development work. This thesis investigates the opportunities and current limitations of cloud-based design automation in the context of Building Information Modelling. A web application prototype that performs selected functionalities in BIM models was developed as part of the thesis to validate the defined implementation workflow for cloud-based design automation systems using the selected implementation approach. Furthermore, the evaluation of the prototype helped to analyse the opportunities and current limitations of cloud-based design automation systems in daily BIM workflows.

Contents

List of Figures

List of Tables

List of Abbreviations

AEC	Architecture, Engineering, and Construction
AI	Artificial Intelligence
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BIM	Building Information Modelling
CAD	Computer-Aided Design
CDE	Common Data Environment
CSS	Cascading Style Sheets
DOM	Document Object Model
FM	Facility Management
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ICT	Information and Communication Technologies
IDE	Integrated Development Environment
IFC	Industry Foundation Class
JSON	JavaScript Object Notation
KBE	Knowledge-Based Engineering
MOKA	Methodology & tools Oriented to Knowledge-based engineering Applications
MPA	Multi-Page Applications
PaaS	Platform as a Service
REST	Representational State Transfer

ROA	Resource-Oriented Architecture
SaaS	Software as a Service
SME	Small and Medium-sized Enterprise
SOA	Service-Oriented Architecture
SPA	Single-Page Applications
SVF	Simple Vector Format
UI	User Interface
URL	Uniform Resource Locator

1 Introduction

1.1 Motivation

The Architecture, Engineering, and Construction (AEC) industry has experienced tremendous digitalisation in the last few decades. Building Information Modelling (BIM) has played a significant role in this digital transformation and is certainly the epitome of digital achievements in construction. Building Information Models (BIM) allow architects, engineers, interior designers, manufacturers, and project managers to communicate in real-time and with unparalleled precision on even the most complicated projects (Borrmann et al., 2015). Given the amount of Small and Medium-sized Enterprises (SME) in the AEC industry, the implementation of BIM in SMEs is essential in this digital transformation of the industry.

One of the main challenges in adapting BIM in SMEs is the need for high initial investment. The SMEs might not have the budget to buy expensive modelling software and do necessary training and upskilling, which could make them stand out of the BIM process. Also, it is not convenient to always have available CAD software to update design changes during design developments, especially when project partners work remotely. So, a location independent, central accessible cloud-based platform, where all stakeholders could not just view and extract information, but the possibility to edit and update BIM information could make the BIM process more transparent and convenient with better collaboration and coordination among stakeholders.

Cloud – BIM integration is considered to be the second generation of BIM development (Wang et al., 2014). Emerging technologies like cloud computing and Software-as-a-Service (SaaS) can solve the existing problems such as high initial investment due to expensive license requirements of on-premise modelling software and their extensive hardware requirements, scalability and accessibility issues, and the standalone nature of traditional design and engineering approaches. It can be seen that cloud-based platforms and products are getting high acceptance in the current market due to their convenient, affordable and user-friendly nature (Elban, 2020). The high-performance computing capabilities provided by cloud computing could be utilised to outsource computationally intensive tasks to the server. Furthermore, the automation of the time-consuming, repetitive design tasks in daily BIM workflows frees up time for more

creative development work for various stakeholders. The fast execution times, universally accessible web interface and customizable nature make cloud-based design automation a preferred choice for various stakeholders, especially those limited by the computational power and initial investment.

1.2 Objective

The thesis aims to investigate the opportunities and current limitations of cloud-based design automation in the context of Building Information Modelling. Also, this thesis discusses the importance of cloud computing and SaaS structures in the AEC industry. In the course of the work, a prototype web application will be developed that tests some cloud-based design automation systems in various complexities to validate the prospects of model updation and design automation tasks. The necessary architectures, protocols, and data exchange systems needed to interact with server-based APIs, along with the chosen implementation approach, will be discussed. Besides the functionalities, the applicability of the implementation will be verified on example use cases, and the challenges it implies over manual modelling will be identified. The prototype targets various stakeholders in the building lifecycle, easing the restriction of good computing capabilities and initial investment, getting information easily and quickly, updating the model and performing automation tasks.

1.3 Structure

Chapter 2 provides an overview of BIM, its current state of adoption and its evolution to cloud-BIM. This is followed by chapter 3, which explores the significance of emerging technologies such as cloud computing and SaaS in the AEC industry. Then the cloud-based design automation aspects are discussed, along with the applications it could provide. In chapter 4, the technical foundations required for implementing a prototype web application that could test the cloud-based design automation systems is described before introducing the actual development, user interface and functionalities in the chapter after (Chapter 5). Chapter 6 deals with the evaluation of the prototype, considering some of the use cases and end by summarising the gained insights and outlining future improvements.

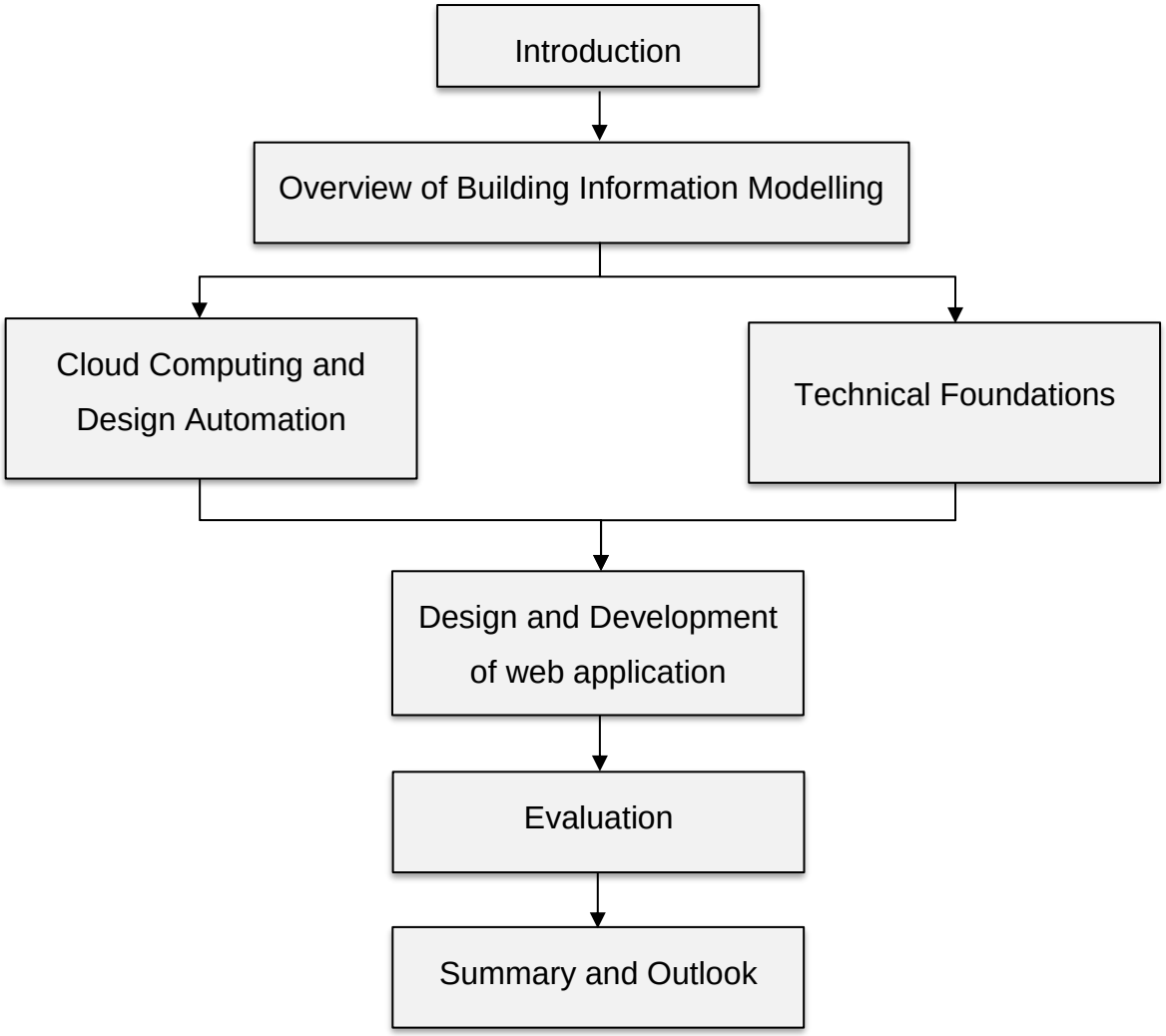

Figure 1: Structure of the thesis

2 Overview of Building Information Modeling (BIM)

Definition

A building information model is a digital representation of a built facility with a great depth of information. In addition to the three-dimensional geometry, the building information model also consists of semantic information such as material information, costs, technical properties, manufacturer data, etc. The term BIM is not only limited to the digital representation of the building, but it also encompasses the processes of creating, maintaining and updating the BIM model using software tools across the whole lifetime of the building (Borrmann et al., 2018).

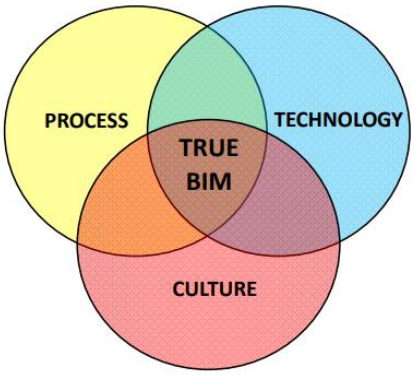

Figure 2: BIM as the combination of technology, process, culture (Macdonald, 2011)

As per the US National Building Information Modelling Standard, BIM is defined as (NIBS, 2012):
"BIM is a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle, defined as existing from earliest conception to demolition. A basic premise of BIM is a collaboration by different stakeholders at different phases of the life cycle of a facility to insert, extract, update or modify information in the BIM to support and reflect the roles of that stakeholder."

In short, BIM can be defined as "a modelling technology and associated set of processes to produce, communicate and analyse building models" (Eastman et al., 2008).

2.1 Evolution of BIM

The concept of BIM is not new. It has been in development since the 1970s, but the term Building Information Modelling was used for the first time in 1992 by van Nederveen and Tolman. The software tools for modelling buildings were first introduced around the 1980s, such as Building Description System from Chuck Eastman (Eastman, 1974). Since then, many software products from various vendors with BIM functionalities have been introduced. Today, BIM software products such as Autodesk Revit, Graphisoft's ArchiCAD, Vectorworks Architect etc., assist designers, planners, and constructors, to design and visualise the building process before its construction (Sacks et al., 2018). Using such BIM applications, intelligent three-dimensional models are created by linking them with semantic information like physical and functional characteristics, which act as a primary source in planning, executing and operating buildings using a collaborative approach. These models are being used by architects, planners, engineers, and contractors to coordinate the work and communicate more effectively for better collaboration.

Even though the data in the BIM is digital, it might not be interoperable and in exchangeable form as the BIM software used to author the building models might use proprietary data structures in their software. This interoperability issue hinders the collaboration and data exchange between various stakeholders as each stakeholder might work on separate BIM software and is regarded as an obstacle to BIM adoption in the industry. This poor software interoperability could be resolved by "big open BIM" approach for efficient collaboration within BIM stakeholders and processes. The term "big" represents the extend of BIM usage and involves the BIM information and model used among various stakeholders across the entire lifecycle of the building. The term "open" indicates that even though the software used is different, vendor-neutral data formats such as IFC[1] are utilised for the exchange of data (Borrmann et al., 2018).

Lately, the use of BIM and digital tools in the AEC industry has been increasing, resulting in increased productivity, high product quality, making the designing, constructing, operating buildings more effective and efficient. However, the application of digital information in the AEC industry is still lagging compared to other industries.

[1] IFC stands for Industry Foundation Classes
URL: https://technical.buildingsmart.org/standards/ifc/

According to the McKinsey report, as shown in Figure 3, construction ranks near the bottom in digitisation compared with other industries.

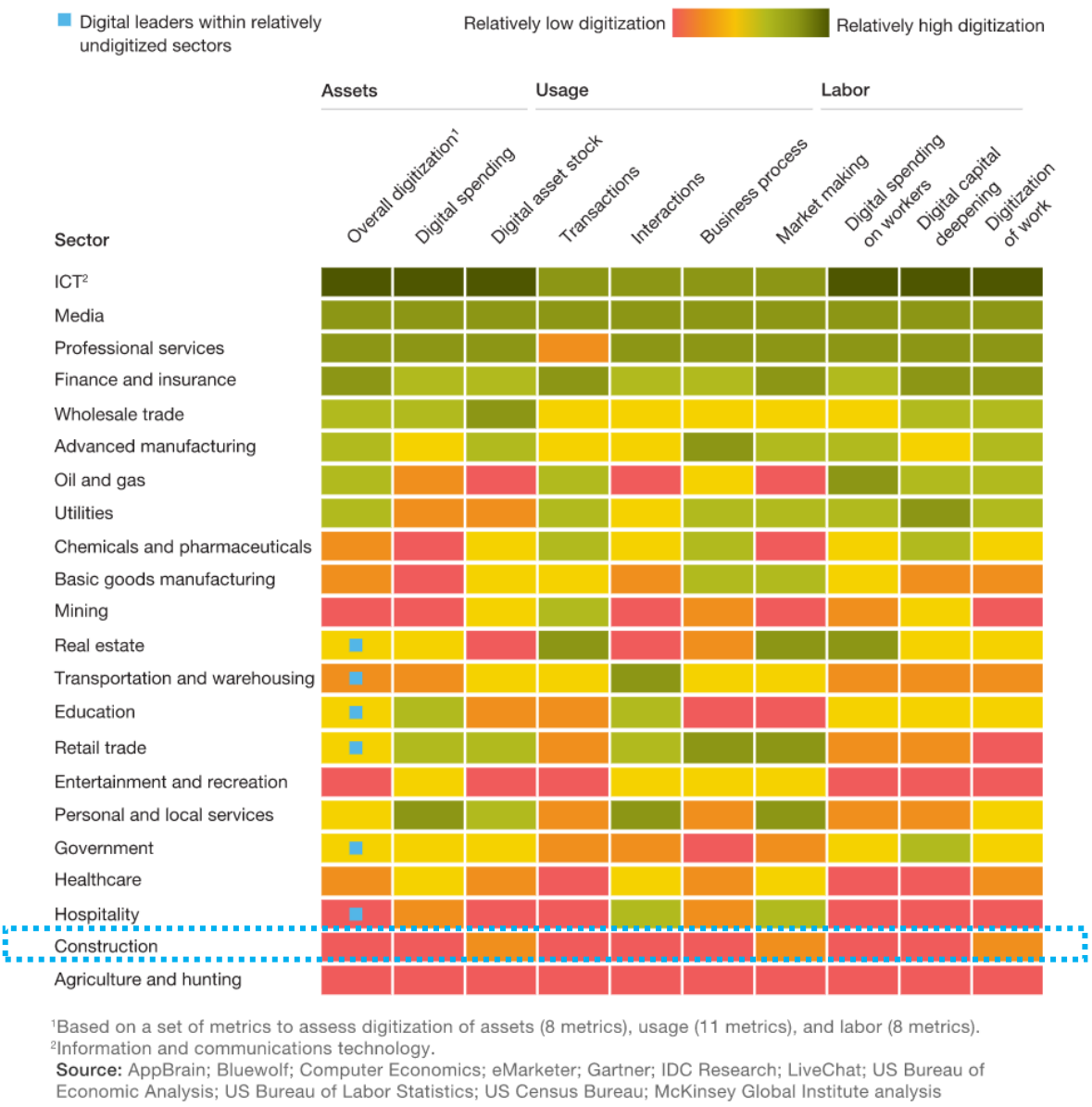

Figure 3: McKinsey Global Institute Digitisation Index (Mckinsey, 2019)

The information is still mainly being transferred using conventional methods in the form of drawings - technical drawings either by physical means printed in paper or digital but in a limited format - the 2D drawings created using digital software. This lack of digitisation results in the loss of valuable information during the handover between various phases across the entire building life cycle (see Figure 4).

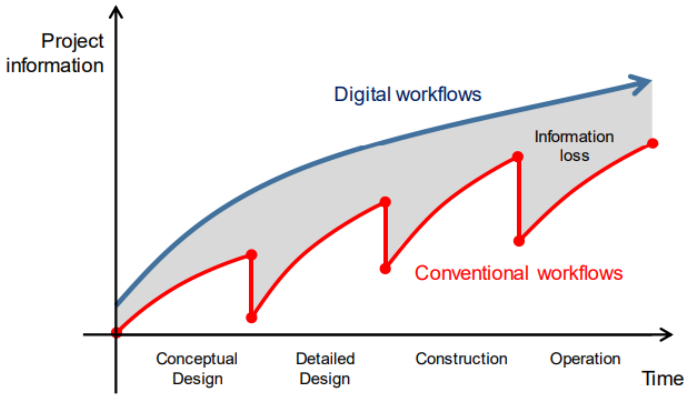

Figure 4: Information loss in conventional and digital workflows (Eastman et al., 2008)

2.2 BIM Maturity Levels

The concept of an entirely digitised model-based working (big open BIM approach) in the AEC industry is still not realised and is what the scientific community has been researching on. The AEC industry still lags behind in the BIM implementation, and appropriate measures have to be taken to introduce the novel technologies, innovations and changes in BIM. The BIM maturity model developed by the UK BIM Task Group describes various levels of shared collaboration in a construction project known as BIM maturity levels. These discrete levels are a measure of the maturity of the construction supply chain's ability to exchange information and collaborate.

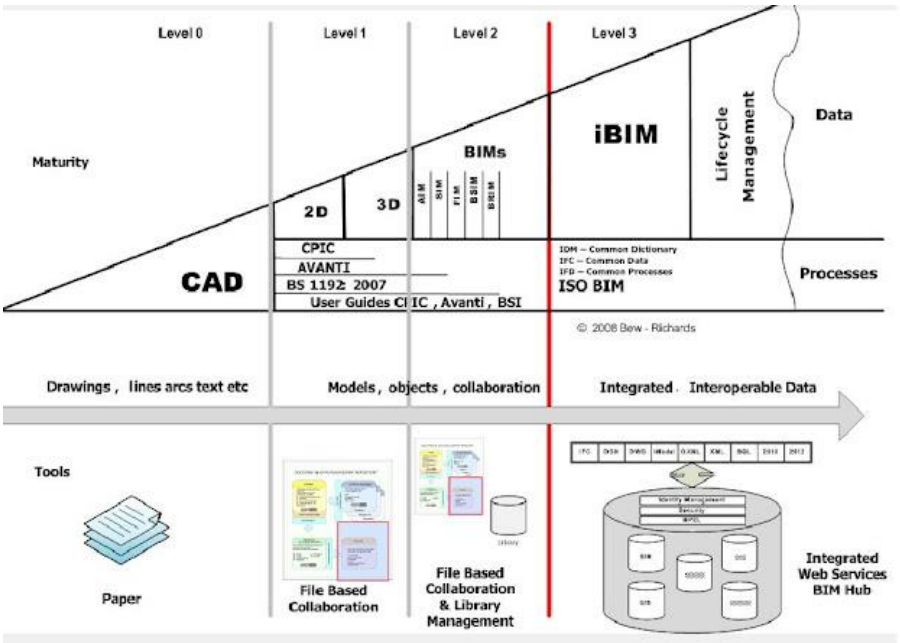

Figure 5: BIM maturity levels (Bew & Richards, 2021)

Borrmann et al. (2018) describe various maturity levels defined by the UK BIM Task Group as follows:

Level 0: It is the conventional working practice based on Computer-Aided Design (CAD), including 2D drawings and the exchange of information using traditional paper-based drawings or, in some instances, digitally using PDF. It is the entry-level of BIM where there is effectively no collaboration. Moreover, it is the first step of using digital tools in the AEC industry and is rarely used nowadays.

Level 1: It is a combination of 2D and 3D, where most of the design would still be realised by 2D drawings and includes partial 3D modelling of the building. Level 1 focuses on the transition from CAD to 2D and non-federated 3D models, which can also be described as 'Lonely BIM' as it does not involve sharing models between various stakeholders. This level involves partial collaboration among project members as each creates and manages their own data and, through standardisation of the information and models, exchanging data by sending and receiving individual files.

Level 2: It involves using BIM software tools for federated digital 3D models, where each discipline develops its model. In this level, 2D drawings are generated from BIM models. The exchange of information and collaboration with each project stakeholder involved in the construction project is realised using files (models in native formats or common file types like IFC) managed in a central platform called Common Data Environment (CDE). Collaborative working is an essential aspect at this level and requires timely information exchange and good coordination among various stakeholders and systems. Level 2 has been mandated for the public projects by the British Government since 2016, which resulted in most of the companies moving from Level 1 to Level 2.

Level 3: It is the ultimate goal of the construction sector, which aims to obtain complete integration of information in a cloud-based environment. This level comprises a single, collaborative project model saved and maintained in a cloud server and accessible to each stakeholder, who can add, update and manage their information in the central model during the entire lifecycle of the building. This level is often referred to as integrated BIM (iBIM) and is based on the implementation of big open BIM.

BIM Level 2 focuses more on the design phase, and models created at this BIM level are exported and imported to disconnected systems, causing errors, rework, and version control issues. Even though the information created by the design team are

exchanged to some extent using CDEs, it does not flow seamlessly to the following stages in the construction project. Furthermore, there are no integrated systems in BIM Level 2 to use the potential of BIM data entirely, and even some stakeholders are removed from fully collaborating on the BIM model. These issues are resolved by BIM Level 3, which connects the information chain from start to end of the entire building life cycle. Using BIM Level 3, a single source of truth is established accessible by all stakeholders through web services. It allows the data to be transferred seamlessly not only among design members but also in construction, facility management, enabling full collaboration and building lifecycle management (Dassault Systèmes, 2021).

2.3 Cloud-based BIM

Recently, with the increasing popularity of cloud technologies, much research has been conducted in integrating cloud aspects in BIM. It has been found that cloud-BIM could greatly benefit the AEC industry and is expected to create another wave of change after the beginning of BIM (Wang et al., 2014).

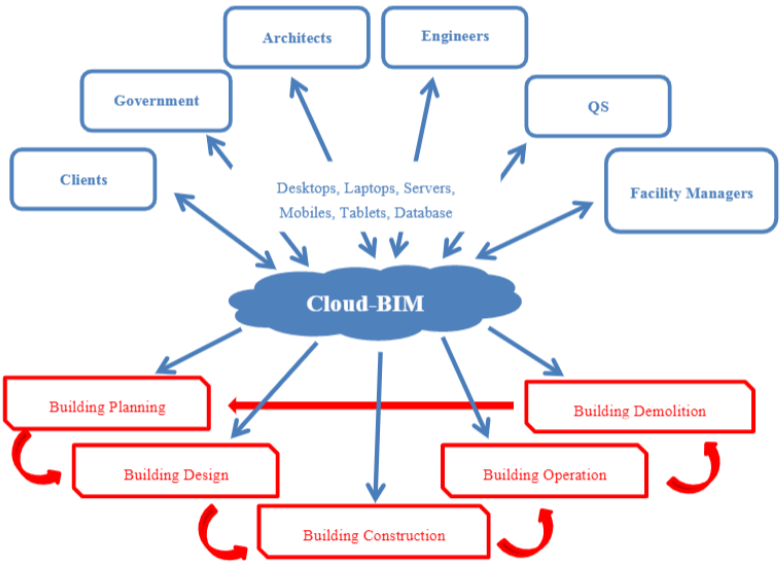

Figure 6: Cloud-BIM concept (Wang et al., 2014)

Advances in cloud-BIM have led to a simplified exchange of data, storage and real-time collaboration among various clients through the project life cycle. According to Afsari et al. (2016), cloud-BIM is found to promote collaborative BIM data generation and consumption. In addition, cloud-BIM offers an environment for easy file management and quick access to BIM documents and models and high-performance

computing abilities, which makes it beneficial for managing and storing BIM data (Alreshidi et al., 2018). Furthermore, cloud-BIM could benefit from the distributed storage of large BIM data across multiple servers for parallel analysis and computing, along with the visualisation of all online BIMs through web-based services (Chen et al., 2016). These technologies have emerged not just to resolve challenges with collaboration among building project partners, but also to utilise the cloud computing abilities to perform several functions, which are looked upon further in the following chapter. Despite the benefits, cloud-BIM faces some challenges, including interoperability of cloud-BIM services, administration and governance, a lack of proper expertise, and reliance on internet connectivity. Concerns about cloud-based BIM administration and governance are a big problem; especially when all consultants have access to the BIM models, regulations about who owns the models and who verifies them for quality must be carefully established. Losing data and work sets during the upgrade was also identified as a problem. As cloud-BIM promotes the full collaboration among project participants over the complete lifecycle of the building, along with easy access of data saved in a central server, it helps to achieve what BIM Level 3 envisions.

In order to get the necessary background for the investigation of the research topic ‘Cloud-based design automation in the context of BIM’, a cluster on cloud-BIM, as shown in Figure 7, was formed with key topics and are further investigated in detail in chapter 3.

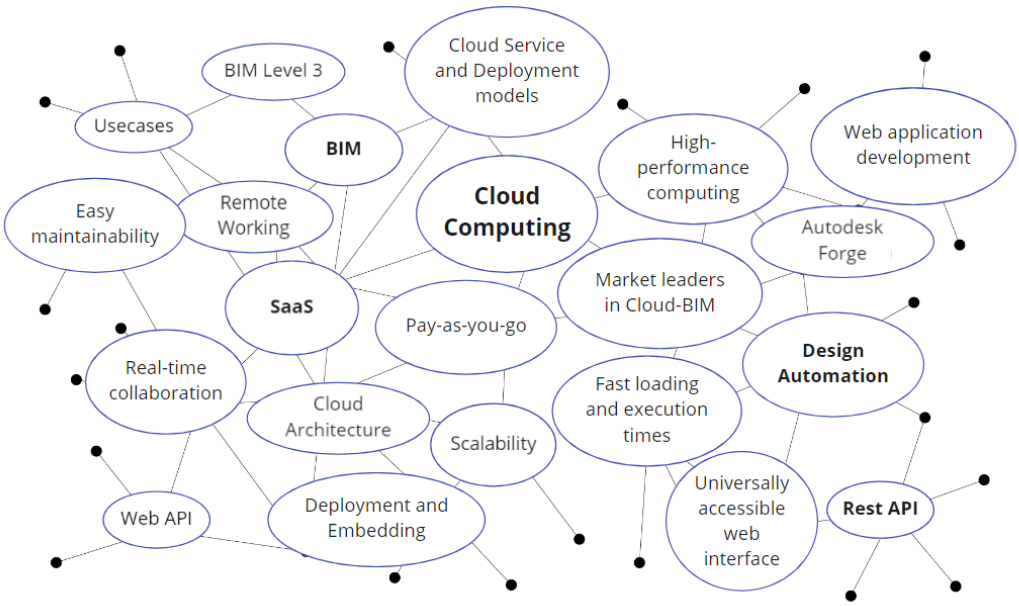

Figure 7: Cloud-BIM cluster

3 Cloud Computing and Design Automation

This chapter deals with cloud computing and SaaS, the emerging technologies in the AEC sector. In addition, a brief review on how Cloud computing and SaaS can help the cloud-based BIM aspects to solve the limitations of BIM is addressed. Finally, cloud-based design automation and its various applications are discussed.

3.1 Cloud Computing

Cloud computing delivers computing services such as servers, storage, networking, databases, software, and high-performance computing over the internet (Microsoft Azure, 2021). It enables people to move from existing preconceived notions about computing. Cloud computing is a rapidly increasing technology that can be used by a wide variety of devices such as computers, tablets, smartphones, removing the barrier of technology investment. The term cloud computing has been there for some time, but the term really became popular since 2007 when Google and IBM announced a collaboration in this domain (Vouk, 2008). According to the US National Institute for Standards and Technology (NIST): "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell & Grance, 2011). The basic concept behind cloud computing is to manage and schedule uniformly network-connected computing resources to create a pool of computational resources that can deliver a user service tailored to their own needs and requirements. A "cloud" is a network that serves as a resource provider, and these resources can be extended further, accessible at any moment and paid just for the use.

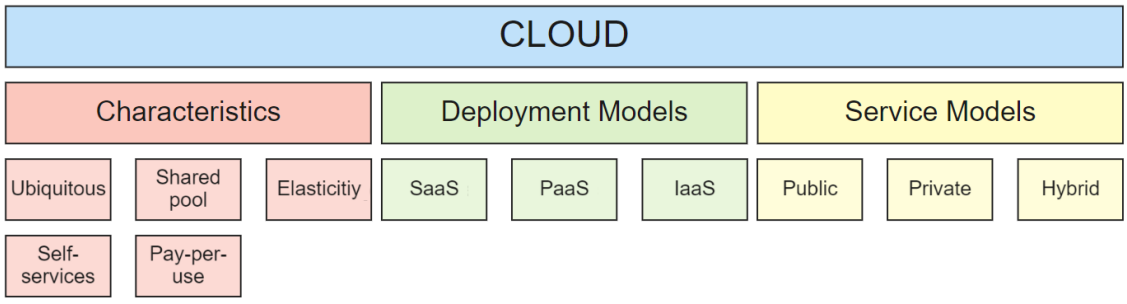

Figure 8: Overall view of cloud computing

Cloud computing architecture:

The cloud computing architecture consists mainly of four layers, as shown in Figure 9: hardware/datacenter, infrastructure, platform, and application. The hardware layer is responsible for managing the physical resources of the cloud, such as physical servers. The infrastructure layer, also known as the virtualisation layer, manages computing resources by partitioning physical resources using virtualisation technologies. The platform layer includes operating systems and application frameworks used to reduce the burden of deploying applications in the virtual machine. Finally, the application layer, which is the top layer, contains actual cloud applications. These layers are loosely coupled with each other, which allows each layer to evolve separately (Zhang et al., 2010).

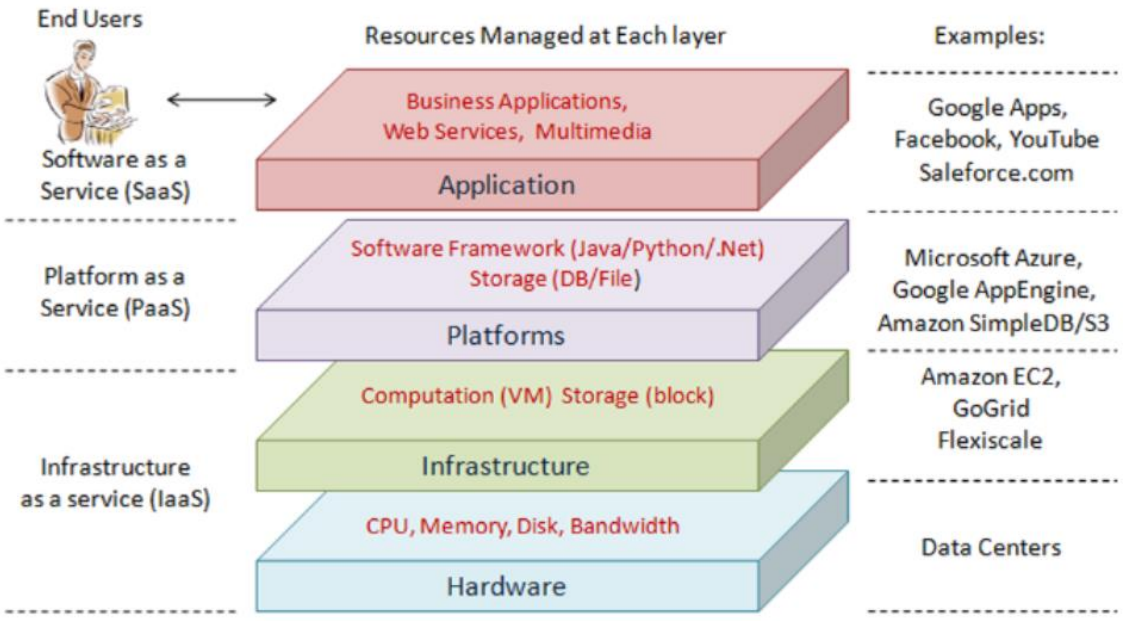

Figure 9: Cloud computing architecture (Zhang et al., 2010)

Service models:

Cloud computing can be categorised into three main types to deploy the cloud services: public, private and hybrid cloud. Public clouds are clouds in which third-party cloud service providers such as Amazon Elastic Compute Cloud (EC2), Microsoft Azure, and Google Cloud deliver cloud services to the general public. A private cloud is designed for the computing resources to be used by a single business or organisation. It is usually built and managed by the organisation itself on a private network. Hybrid clouds

combine public and private clouds and allow the data and applications to be shared between them. Hybrid cloud offers more flexibility and deployment options than public and private clouds (Microsoft Azure, 2021).

Deployment models:

According to NIST, cloud computing services can be grouped into three broad categories, also known as services models (Mell & Grance, 2011).

1. **Software as a Service (SaaS**): SaaS is a service that delivers software applications over the internet on-demand. It allows accessing the applications via an internet browser rather than installing and running on a desktop PC or business network. The difficulties caused by running applications in a traditional way using computers such as downloading, installing, managing applications like software upgrades are handed over to cloud providers that host and manage these software applications and their underlying infrastructure.
2. **Platform as a Service (PaaS):** PaaS is a service that provides end-users with a platform to develop and manage their own software applications. It allows creating web or mobile apps without the hassle of setting up and managing underlying cloud infrastructures such as servers, storage and databases.
3. **Infrastructure as a Service (IaaS):** IaaS is a service that delivers just the infrastructure needed for application development. The cloud providers provide the infrastructure such as servers, storage, networks, operating systems and virtualisation on a pay-as-you-go basis.

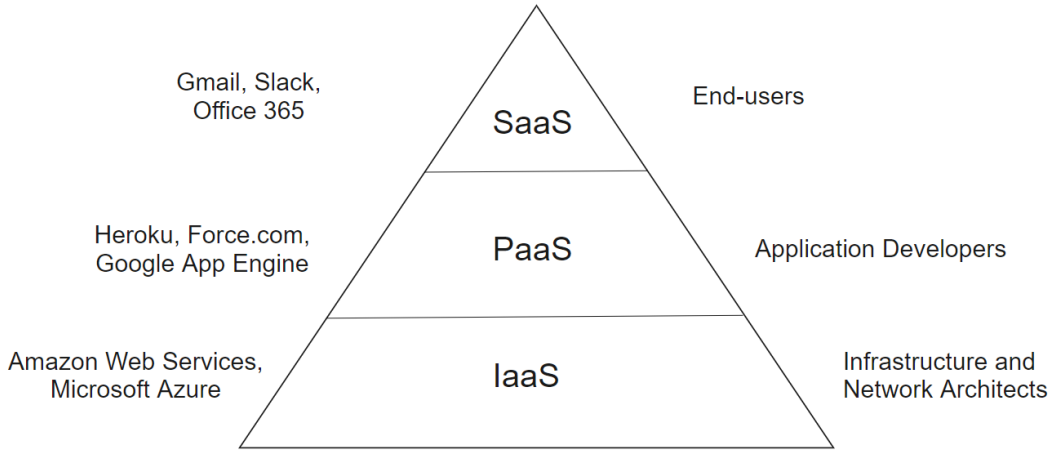

Figure 10: Cloud computing service models

Uses of cloud computing services:

Instead of creating their own IT infrastructure, businesses can use computing resources offered by third parties on the internet. Since cloud services are delivered over the internet, businesses must have access to the internet to use them, which about 98 per cent of EU businesses with ten or more employees do in 2020. Among these, cloud computing was only used by 36% of people in 2020. The vast majority (76 per cent) of businesses that reported utilising cloud computing chose a cloud solution to host their email systems. Two-thirds of respondents utilised the cloud to store files, 58 per cent for office applications (such as word processors and spreadsheets), and 47 per cent for hosting databases. The enterprises also used the cloud to access more advanced end-user software programs like financial/accounting (45%) and customer relationship management (27%). In addition, about a quarter of these enterprises utilised the high-performance computing capabilities of cloud computing platforms to operate their own business software applications (Eurostat, 2021).

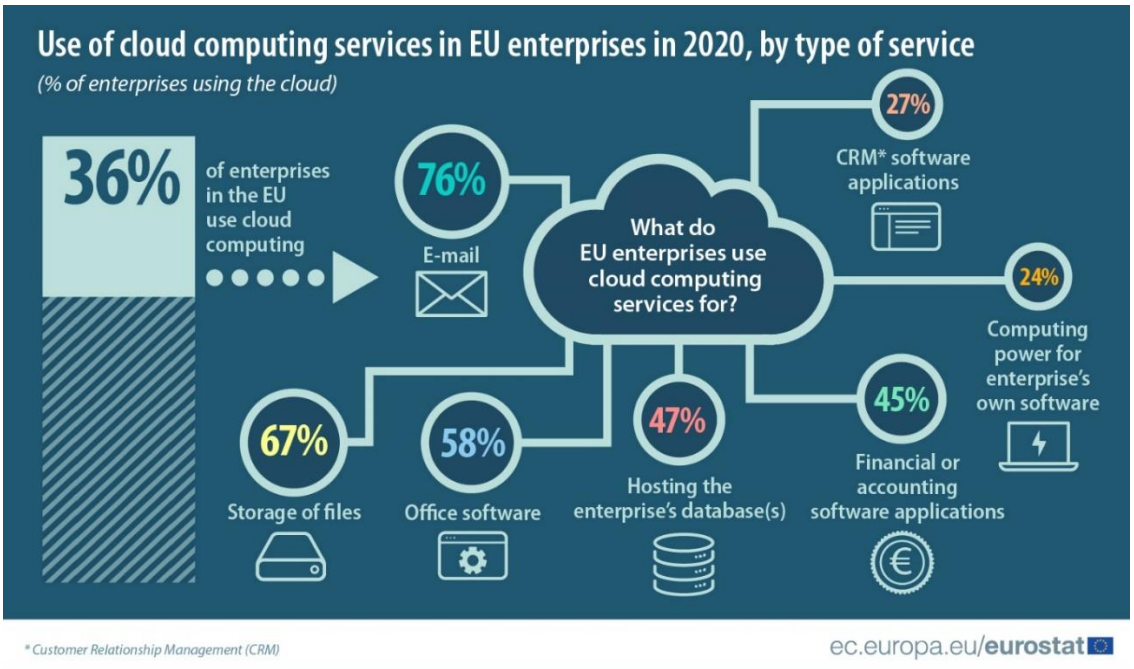

Figure 11: Use of cloud computing services in EU enterprises, by type of service (Eurostat, 2021)

3.2 Cloud Computing in AEC

Cloud computing has gained popularity and is making unprecedented changes across communication and business disciplines. However, the AEC industry is more investment intensive, making it difficult to experiment with new technology. Even though cloud computing has enormous potential in the AEC industry, such applications are not widespread (Oesterreich & Teuteberg, 2016). The application of cloud computing in the AEC industry is an emerging technology and brings several benefits from adopting cloud computing technology in the industry.

Cloud computing can improve resource utilisation, save costs and implement a distributed management model suitable for solving the existing data sharing problem between various stages and stakeholders in BIM. Porwal & Hewage (2013) suggested that the pre-planning of sustainable construction across the building life cycle can be improved by applying cloud computing with their BIM partnering framework. Zheng (2018) explored the application of cloud-BIM in the collaborative design phase, suggesting that cloud computing technology can collaborate among various disciplines and achieve improved design, project efficiency and visual 3D dynamic preview and rendering. Redmond et al. (2012) investigated the demand and future problems of applying BIM-based cloud computing integration platform through expert interviews and discussed how to use cloud-BIM for information exchange. Ma & Sacks (2016) presented a cloud-based BIM platform for information exchange based on MongoDB, a cloud-based database to achieve the storage and sharing of IFC format data. This research addressed the problem of having a specialised internal data schema for various BIM authoring tools and proposed a cloud-based platform where users can query and enrich model objects.

Du Juan & Zheng (2014) explored the system architecture of cloud-BIM and developed a framework to deploy a hybrid cloud-based BIM application, which provided information sharing and interoperability among various stakeholders and different project phases. Jiao et al. (2013) investigated the application of cloud computing in BIM for lifecycle data management. A cloud-based BIM tool was developed and used during the construction phase and concluded that such a system could facilitate the construction process and suggested its usage in lifecycle data management. The increasing interest in the application of cloud computing in BIM was visible in the literature analysis and is found to be an active area of research.

Cloud computing is being used at several project phases in a building that includes planning, design and construction but is currently not much used in handover and operations phases. However, cloud computing in BIM could also be extended to the facility management and operations domain so that it could provide quick and real-time access to BIM data, manufacturer and building maintenance records, real-time data management on existing buildings and so on (Wang et al., 2014). Furthermore, Wang et al. (2014) found that cloud BIM technology, independent of location, would allow real-time construction tracking, collaboration, clash monitoring, and information sharing among construction project team members. Cloud computing also enables the use of other emerging technologies such as the Internet of Things (IoT), Big data, Augmented & virtual reality, mobile technologies in the AEC industry etc. Although these emergent innovations have taken away the attention from cloud computing, cloud computing remains the cornerstone for modern collaborative IT architecture (Bello et al., 2021).

According to Mahamadu et al. (2013), the challenges of cloud-BIM integration includes user access authorisation, boundaries to information sharing, and legal and contract limitations. One of the most significant barriers to cloud-BIM adoption is a lack of clarity about who is accountable for cloud-BIM model ownership (Wang et al., 2014). The key to efficient cloud service execution is interoperability. However, interoperability will be difficult for cloud-BIM solutions developed by software vendors using multiple cloud platforms (Afsari et al., 2016). Another barrier to cloud-BIM adoption is its reliance on internet access, which is particularly difficult in developing nations where access is not always available. Besides many advantages provided by the cloud-BIM, it also has some challenges affecting cloud-BIM integration, which should be further researched.

3.3 Software-as-a-Service (SaaS) in AEC – State of the art

Compared to a traditional on-premises solution, a BIM software provided as software-as-a-service offers the user a variety of benefits. The user does not need to worry about installation, upgrades and maintenance of the software as it is handled by the software manufacturer. BIM SaaS software tools can be used easily and are scalable on a pay-as-you-go basis. Furthermore, such BIM software applications are accessed online where the user can log in to the BIM tools with their account using a web browser. BIM authoring and analysis applications that are typically on-premise software require processing power, and with cloud computing gaining popularity, these

requirements can be partially met by cloud-based services and rented by the hour for as long. BIM viewing, mark-up, and commenting are increasingly being managed through web-based applications.

SaaS was mainly introduced to AEC to solve the increasing data requirements of BIM models, which paved the way to cloud-based BIM collaboration platforms. In June 2011, Autodesk released Autodesk BIM 360, a new generation BIM solution that moved BIM procedures to the cloud and delivered professional cloud services to AEC professionals. This cloud-based construction management platform enhances project delivery and outcomes by connecting different teams and data in real-time, allowing project members to anticipate, optimise, and manage all aspects of the project (Autodesk, 2021a). Similarly, Graphisoft released BIMCloud for team collaboration for all disciplines to work together efficiently in building projects. This solution acts as a framework for improving interaction and cooperation between stakeholders and central file storage, document and version control tools (Graphisoft, 2021). Furthermore, Allplan provides BIMplus for the same purpose where BIM model data, information, documents and various tasks are managed centrally throughout the whole building life cycle (Allplan Bimplus, 2021). Likewise, Bentley offers ProjectWise as project delivery and collaboration software that extends and improves project communication among various building project participants (ProjectWise Web, 2021).

Autodesk released AutoCAD Web, an online CAD product with an improved cloud service, in September 2011. This web-based application allowed users to create and modify CAD files online, as well as do basic tasks including drawing, editing, and labelling. Also, Autodesk offers FormIt, which is a user-friendly 3D sketching tool for the design stage. It provides a simple and powerful design environment and allows users to design anywhere, at any time and extends the functionalities to have site context, solar- and energy analysis (Autodesk FormIt, 2021). Insight 360 is a web-based SaaS offering from Autodesk for energy modelling and simulation for energy efficiency, thermal comfort, photovoltaics and lighting (Autodesk, 2021b). Nova AVA is a web application first released in 2014 for model-based construction cost management as a SaaS service. Using Nova AVA, cost planning, quantity takeoff, cost analysis, bill of quantities etc., are possible, and single components are linked with the 3D model using the IFC format for the model-based working (Avanova, 2021). MTWO is a cloud-based construction enterprise software platform provided by RIB Software that allows managing the projects in 5D BIM from start to finish. It connects all

stakeholders to collaborate at every stage of the process, on any device, at any time and could be integrated with 5D and 6D aspects of BIM Dimensions along with the Internet of Things (IoT) and Artificial Intelligence (AI) (RIB Software, 2021).

By using Autodesk Forge, a cloud development platform from Autodesk, innovative, cloud-based SaaS applications can be created for specific purposes, which is further explored for prototyping purposes over the course of the Master thesis.

3.4 Cloud-based design automation

“The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency ” - Bill Gates

To automate any process, workflow and task, it should be understood and checked whether it is valuable or not. Activities that are routine, repetitive and time demanding are usually more suitable for automation since they usually handle explicit information and knowledge and, therefore, are easier to formalise than more tacit information and knowledge. Computational intelligence, more numerical or soft computing approaches such as neural networks, evolutionary algorithms, fuzzy logic, is one way to handle more tacit, intangible information (Hopgood, 2001). According to Stokes (2001), knowledge availability, organisational readiness (IT maturity), hardware and software availability is essential to be considered before choosing to automate.

3.4.1 Automation in the AEC industry

"Automation is the technique, method, or system of operating or controlling a process by highly automatic means, as by electronic devices, use of control systems and information technologies, reducing human intervention to a minimum" (WordReference, 2021). While automation can be defined in various aspects, it can be mainly classified into three primary opportunities. The first is where automation is applied on-site for physical tasks, mainly dealing with robots and machines such as robots laying bricks. The second opportunity arises from the automation of modular construction or production in factories such as 3D printing of components. Finally, the third opportunity focuses on digitisation and automation of design, planning, and management procedures, which is being dealt with over the course of this thesis (McKinsey & Company, 2019).

Automating chains of engineering activities is not a concept; engineers have used it since the early development of computer-aided modelling (Dixon, 1995). Design automation is originated from the automated design of electronic circuits and chips (MacMillen et al., 2000). According to Sandberg et al. (2016), design automation within the construction sector includes various fields such as BIM, master models, knowledge-based engineering, configuration, modularisation, platforms and simulation. Among these fields of design automation, the knowledge-based engineering approach is the field of interest in the context of this thesis.

3.4.2 Knowledge-based engineering

Knowledge-Based Engineering (KBE) allows the computer to perform time-consuming, repetitive design tasks, freeing up time for more creative development work. The computer never gets bored and works faster than the human effort required, and runs everything the same way every time it is supposed to. It is called 'knowledge-based' as the knowledge of engineers is captured, formalised and incorporated into computer-based design automation. KBE has become a norm in the manufacturing industry in automating repetitive design work. It uses modern software techniques to capture and re-use products and process information in an integrated way (Stokes, 2001). For KBE development, MOKA is one of the complete methodologies available (Sandberg et al., 2016). MOKA stands for 'Methodology and tools Oriented to Knowledge-based engineering Applications' and describes ways to capture engineering knowledge to implement it in the KBE application.

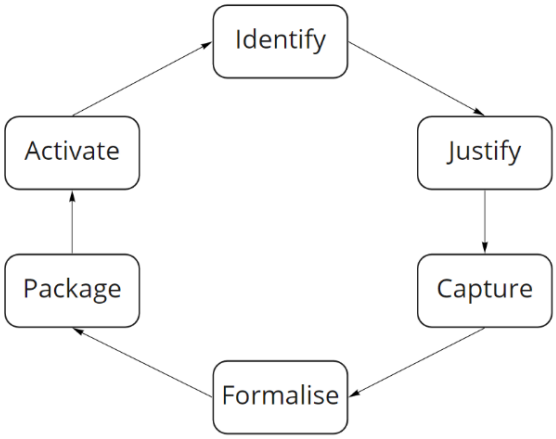

Figure 12: The MOKA phases, adapted from (Stokes, 2001)

The MOKA methodology consists of six steps, as depicted in Figure 12.

1. **Identify**: determines the design automation application's objectives, scope, and a concept level technical specification.
2. **Justify**: investigates commercial, cultural, and technical dangers.
3. **Capture**: gathers raw knowledge and organises it into the informal model.
4. **Formalise**: converts the informal model into the formal model.
5. **Package**: translates the MOKA formal model into code for a KBE application
6. **Activate**: activate the application by distribution, installation, and using the developed application.

According to the research conducted by Sandberg (2015), the MOKA methodology would be well suited for successfully implementing design automation applications in the construction industry. Also, the research discusses the need for a comprehensive methodology for developing design automation applications in the AEC industry, which would help the industry to reuse the existing results.

3.4.3 Evolution of design automation

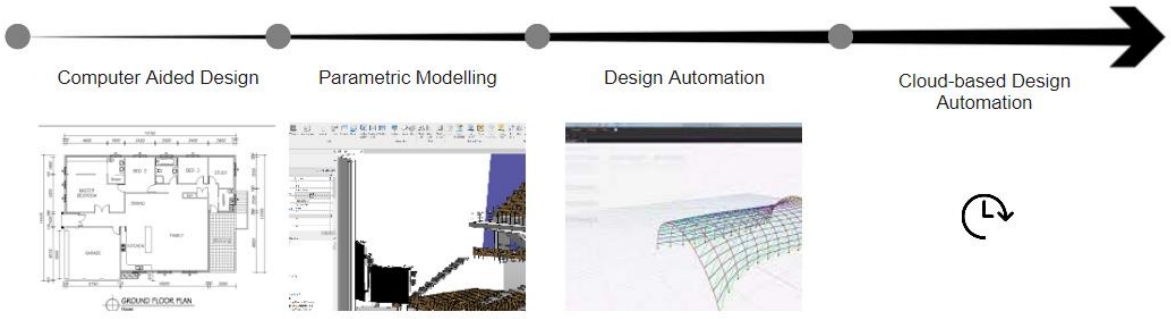

Figure 13: Evolution of design automation

New technological advancements have made the BIM authoring tools more advanced to handle various scenarios of designing buildings digitally. Traditional ways of design include designers using fundamental techniques like sketching to convey ideas from their mind and applying Computer Aided Drafting (CAD) to make them a reality. Autodesk's AutoCAD, SketchUp, ZWCAD and Dassault Systèmes's DraftSight are examples of traditional design tools. In the next level of design, the user defines relationships between traditionally drawn or modelled parts in parametric design. As a result, a change in a single piece of data could cause changes in other pieces and affect the data structure. Such parametric design tools often limit the ability to create

direct relationships, such as a door that depends on a hosted relationship to a wall. Popular parametric design tools include Autodesk's Revit, Graphisoft's ArchiCAD, Bentley's Microstation, Tekla Structures and Nemetschek's Allplan. However, with the advent of design automation, the user can automate tasks within parametric models using automated scripts. The geometry and data are the results of the automatic execution of a set of rules and automated scripts. There are some areas in which the procedures or workflows are dependent on the engineers working with, that could be unique and not so easy to be implemented on a global scale for BIM authoring tools. Such tasks can be automated and thereby assist the design process to be quicker, accurate and saving time. Autodesk's Revit provides scripting and automating capabilities using Dynamo[2] and provide APIs to extend the functionalities using third party plugins or macros in Revit. Similarly, McNeel's Rhino offers scripting and automating capabilities using Grasshopper and also provides API's for external development as well. Such capabilities are found in Nemetschek's Allplan as well and can also be implemented using Generative Components for Bentley's Microstation.

Recently, the Forge development platform by Autodesk started providing functionalities to access these design automation capabilities of the Autodesk environment through web services in the form of design automation APIs. Design automation APIs enable the users to integrate Autodesk's key product engines into their own processes and applications. The cloud-based design automation APIs can access practically most of the capabilities offered by Autodesk's AutoCAD, Inventor, Revit, AutoCAD and 3ds Max. It is described more in detail in section 4.2.1. As the industry moves to SaaS applications, cloud-based design automation can be used to utilise and extend the capabilities of BIM authoring tools online, which could connect business systems to powerful web services to accelerate the work processes and improve the business flexibility.

3.5 Applications of cloud-based design automation in BIM

As the AEC industry embraces its digital transformation, automation in design applications is expanding, making the development process more organised and less time-consuming. A lot of focus has been on design automation in the design phases

[2] Dynamo is a visual programming tool that provides access to Revit API functions.
URL: https://dynamobim.org/

of the construction project. Cloud-based design automation helps create cloud-native applications and services that could create, read, and modify BIM data using the BIM authoring tool engine in the cloud. This enables the automation of workflows and processes that could utilise the high-performance computing capabilities, mobility and low-cost capabilities of cloud computing.

A few possible workflows that could be automated using the cloud-based design automation are identified and are provided below:

- Automation of model creation and documentation
- Extraction and generation of automated reports
- Modification of existing models to maintain company standards
- Explore and analyse model data
- Automatically replace out-of-date content
- Fix common modelling mistakes
- Upgrade models from previous versions to new versions
- Cleaning up data sets
- Model-based design communication and coordination (adding missing information, parameter-based approval, quick design review)
- Connectivity to external sources like excel database and ERP system

As part of this thesis work, to demonstrate the capabilities of the cloud-based design automation, a prototypical web application was developed that could test the functionalities in daily BIM workflows, which aid model refinement tasks, quality improvement, and update design changes as described below:

- Update the parameters to modify the BIM Model and thereby improving the quality.
- Delete the elements.
- Updating the building elements by refining the element with the required element type.
- Explore the possibility to create new building elements at a given location.

The intend of the prototype is to validate the opportunities and current limitations of cloud-based design automation. Even though the possibilities of automation in the cloud are limitless and could be complex, the scope of the implementation is limited to basic operations in modelling such as updating information, deleting, replacing, and

creating, similar to CRUD operations. It is assumed that verifying the applicability of these cases in varying complexities provides a good base for analysing the opportunities and limitations of cloud-based design automation in the context of Building Information Modelling.

4 Technical Foundations

The essential technical and theoretical foundations required for the prototypical implementation are described in detail in this chapter, following the discussion in section 3.5. It outlines the necessary technical aspects of web application development and web service APIs.

4.1 Web application development

Web application development is the process of building a web application, which is an interactive program created using web technologies such as HTML, CSS, JavaScript. The term “web application” refers to any software that runs in a web browser and is created in a browser-supported programming language depending on the browser to render its contents. The web application is used by a team or a single user to perform the tasks over the internet and is accessed through a web browser. They are basically application software stored in a web server that can be accessed directly from the browser over the internet. Web-based applications, in general, have many advantages over traditional desktop applications, including high reliability, high usability, better technologies, shorter time to market, shorter product life cycles, and continuous maintenance (Papadopoulos et al., 2010).

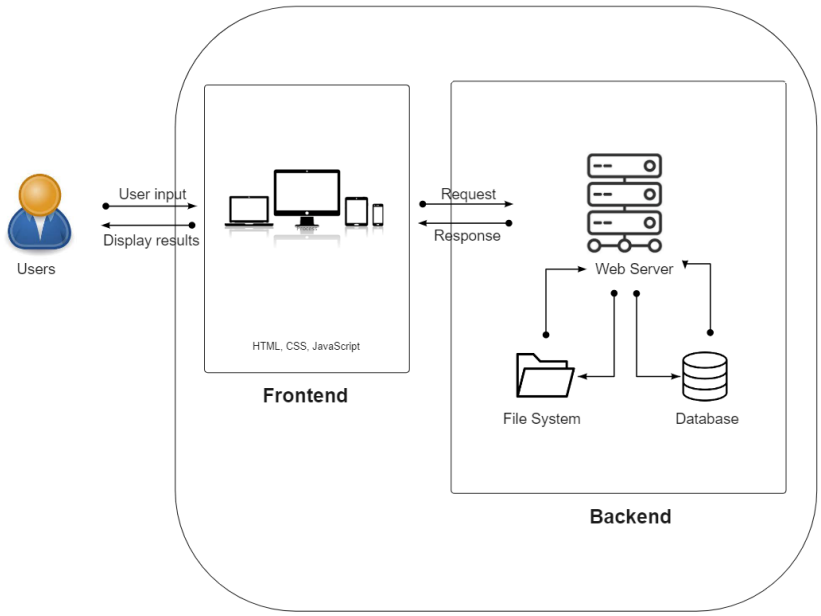

Figure 14: Web application architecture

Table 4-1: Comparing web and desktop applications

	Web application	Desktop application
Installation and updates	• No installation required • Automatic updates by the host software developers	• Manual installation and updates
Platform dependency	• Cross-platform • Portable, mobile and multiple location access	• Platform dependent • Not portable to an extent
Application deployment	• Requires only a browser and an account	• Needs to be installed on the system and have a licensed copy for individual system
Internet connection	• Required • Some web applications can work independently after being loaded	• Not required
Performance	• Dependent on the network and server availability	• Dependent on the system hardware

4.1.1 Traditional web design

A website is the collection of web pages linked together and publicly accessible, typically shares a single domain name and is published at least on a web server. A website is typically accessed using public Internet Protocol networks like the internet and is identified by providing a distinct Uniform Resource Locator (URL) in a web browser. The web browsers render the contents responded by a web server, which usually consists of plain text that has been structured using HTML, styled using CSS, and made interactive using JavaScript.

HTML:

HTML (HyperText Markup Language) is the standard mark-up language for creating web pages. This language was created as the publishing language of the web by Tim

Berners-Lee in 1989 and is maintained by the W3C[3] and the WHATWG[4] (WHATWG, 2021). HTML is the core of every webpage and a basic building block of the web. It defines the meaning and basic structure of the web page. HTML is transferred from the web server while accessing the web page and generates the web page's contents using the provided mark-up. HTML uses mark-up, i.e., HTML tags, to display the text, images and other content in a Web browser. The common HTML tags, also known as “elements” are <head>, <body>, <header>, <footer>, <p>, , <img> and many others. HTML mark-ups are further enhanced by CSS for styling and JavaScript for more interactivity.

```
1.  <html>
2.    <head>
3.      <title> Custom title </title>
4.    </head>
5.    <body>
6.      <a href=”#”> Custom link </a>
7.      <h1> Custom header </h1>
8.    </body>
9.  </html>
```

Code 1 : Simple HTML structure

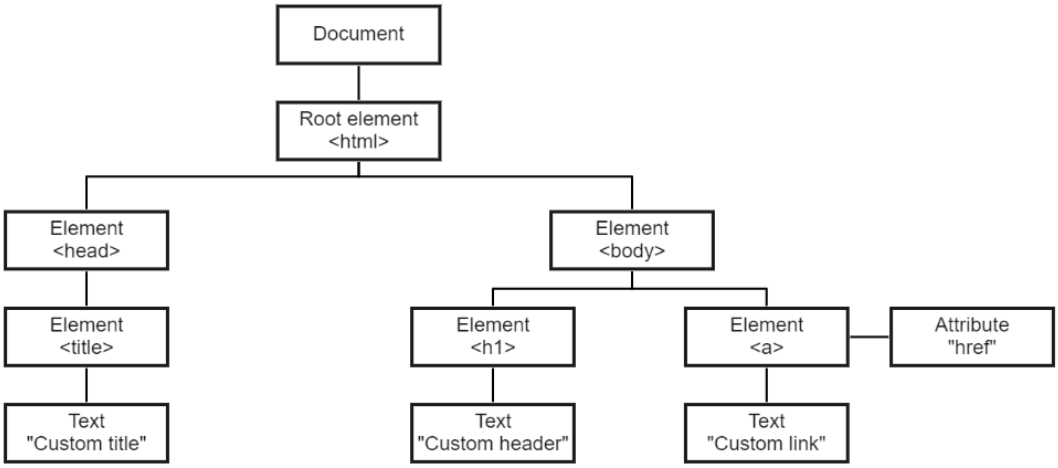

Figure 15: DOM tree

The basic structure of a simple web page is shown in Code 1. Its corresponding Document Object Model (DOM) tree is provided in Figure 15. DOM is the data representation of an HTML document as a tree structure that includes the content and structure of the HTML document on the web. It is a programming interface for web

[3] W3C stands for World Wide Web Consortium (W3C)
[4] WHATWG stands for Web Hypertext Application Technology Working Group

documents used to change the structure, style, and content of the HTML document programmatically. DOM represents the HTML document as objects and associations similar to object-oriented representations, making it possible for programming languages like JavaScript to interact with the web page (Document Object Model, 2021).

Cascading Style Scripts (CSS):

Cascading Style Scripts (CSS) is a style sheet language used in combination with HTML to describe the presentation of a web page. This programming language determines how the HTML elements should appear on the front end of the web page. While HTML provides the structure of the webpage, CSS makes it attractive by styling the content in the HTML document. Figure 16 shows how CSS influences the colours, layouts, and fonts, which are some of the essential characteristics of a web page. It also allows web pages to adapt to various devices and screen sizes. If HTML is the frame of the building, then CSS is like the facade of the building. Typically, these languages are kept separate so that the web pages are built with HTML before they are formatted using CSS, which separates the content and presentation. This approach provides flexibility by creating the same styling for different web pages and creating different styles for the same web page, which benefits the creation of web pages adaptable to various devices and screen sizes.

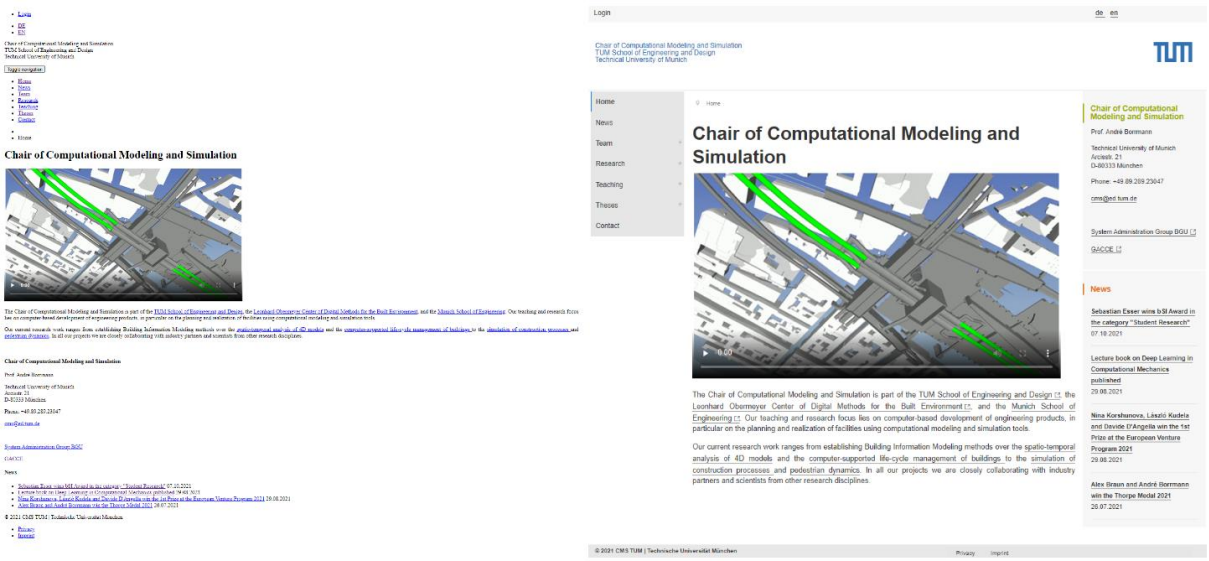

Figure 16: Webpage without CSS (left) and with CSS (right)

CSS has a simple syntax and consists of a list of rules. Each rule consists of a set of properties describing how an HTML element is supposed to look and a selector that identifies the elements of the DOM, the rule should affect.

```
1.  h1 {
2.    color: red;
3.    background-color: blue;
4.    border: 2px solid black;
5.  }
6.
7.  p {
8.    color: green;
9.  }
```

Code 2: Sample of a CSS file

Code 2 shows a basic example of how CSS looks, depicting two different CSS rules. CSS is an essential technology of the World Wide Web alongside HTML and JavaScript, and the CSS specifications are maintained by the World Wide Web Consortium (Cascading Style Sheets, 2021).

JavaScript:

While HTML is used to create the structure of the web page and CSS is used for styling, the JavaScript programming language adds the client-side interactivity to the HTML and CSS elements on the web page. JavaScript is a powerful client-side object-oriented programming language used for any type of web development and is the most complex among the three fundamental web technologies of the world wide web. Nowadays, all modern web browsers support JavaScript, and over 97% of websites use JavaScript to implement the client-side for more powerful and complex functionalities (Web App Development, 2021).

JavaScript makes the web page more interactive and dynamic and alters the structure and layout of a webpage using DOM, thereby improving the user experience. For example, JavaScript can implement scrolling bars and actions to clickable buttons, create call-to-action buttons, confirmation boxes, add new details, and store new information to current information. JavaScript has significantly expanded over time and demonstrated its variability with the evolution of numerous frameworks and libraries – some of which are used to implement the prototype, supported by a large community of developers worldwide.

```
1.  <html>
2.  <body>
3.
4.  <p id=”custom_id”></p>
5.
6.  <script>
7.  document.getElementById(“custom_id”).innerHTML = “Example to input Javascript text!”;
8.  </script>
9.
10. </body>
11. </html>
```

Code 3: Simple JavaScript HTML interaction

Code 3 shows how JavaScript can be used to interact with HTML elements. In order to define the client-side script in HTML, <script> HTML tag is used, and the ‘document’ is the interface representing the web page and gives access to its content, made possible with the provision of DOM. To interact with a specific HTML element using JavaScript, the document.getElementById() method can be used, and in this case, the JavaScript access and write the text "Example to input Javascript text!" to HTML paragraph element with id "custom_id". Similarly, JavaScript can be used to alter the web page content and change the behaviour of the webpage in response to the actions of users.

4.1.2 Types of Web-based Application

Web applications can be developed as Single-Page web Applications (SPA) and Multi-Page Web Applications (MPA), based on the web application architecture. A multi-page web application is a more 'traditional' web design pattern where many web pages are entirely refreshed when there are updates or data changes in them. Whenever there is a data update or data transfer to the server, it causes a new web page to be displayed in the web browser. Multi-page web applications are typically developed for larger systems with many services of different types, preferring more types of interaction with their visitors

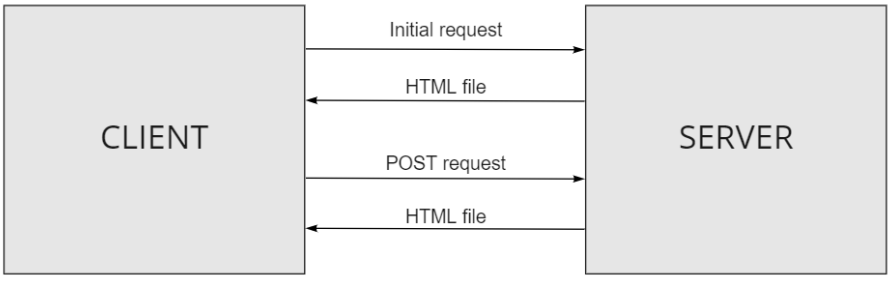

Figure 17: MPA lifecycle

MPA requires many levels of UI when the content is significant. However, the emergence of AJAX[5], a set of technologies to create asynchronous web applications on the client-side, made it possible to create server responses that update the web page without reloading the entire web page. AJAX thus enabled MPA to only refresh parts of the webpage rather than the whole page. This method improved the user experience and eventually led to the advent of SPA (Kaluža & Vukelić, 2018).

Single-page applications are a comparatively newer approach often used to develop simpler applications with less content. SPA is a web application with one page such that a single HTML page is loaded in the browser and is not reloaded during the application life cycle (Mesbah & van Deursen, 2007). Most of the elements of a web page, such as HTML and JavaScript, are loaded once initially on the client-side, reducing the size of data transferred later on, and there is no constant reload of entire web pages every time as in MPA. In case of any requests, SPA dynamically sends the data using asynchronous requests to the server using JavaScript and the new frameworks built upon it, which helps to reduce the network load and takes lesser time. The server returns only the payload[6], not HTML, reducing the size of data transferred. Thus, the required part of the web page is loaded dynamically by manipulating DOM without refreshing the entire HTML page and also gives the user the feeling that the webpage has changed.

As most of the web page's content is already loaded initially and does not have to reload the web page after each update, SPA speeds up the application and simplify the user experience to provide a more 'desktop application like' feeling and makes it possible to work even offline for an extended period of time. This makes SPA the best approach for the prototype web application. Most of the web applications used in daily life are single-page applications such as Gmail, Google Maps, GitHub, Facebook, Twitter, etc.

---

[5] AJAX stands for Asynchronous JavaScript and XML
URL: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started

[6] Payload is the key information in a data block that is being sent or received from the server while making API requests. JSON (JavaScript Object Notation) is a lightweight data-interchange format, one of the most widely used varieties of payloads.
URL: https://www.json.org/

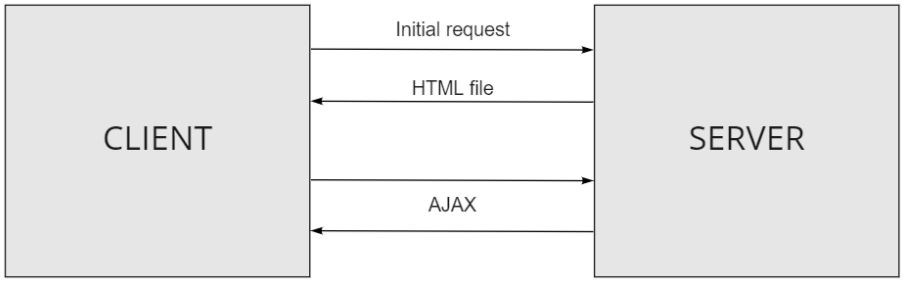

Figure 18: SPA lifecycle

4.1.3 Building a Web Application

In the concept, the advantages of creating a web application rather than a typical desktop application are explored. The users will be expected to enter a web address in the web browser to log in and update the BIM model. Therefore, this section focuses on how to build such a web application.

The development of such a web application is mainly divided into two parts. First is the server-side, popularly known as back-end, which provides the server and will listen to Hypertext Transfer Protocol (HTTP) endpoints and return the information according to the requests. Second, the client-side, popularly known as front-end, is what runs on the users' browsers, providing the correct requests using HTTP endpoints to the server. In short, the front-end provides visual information through the browser, while the server part processes and stores data.

Front-end

Front-end or client-side is the part of the web application what the users see and interact with. Front-end web development involves programming the user interface of the web application, and it is important to note that even though front-end development deals with the interactive and visual aspects, it differs from web design. Web design mainly deals with the look and feel of the web application, whereas front-end deals with the functionalities that convert such web designs to a web application. HTML, CSS, and JavaScript are the web technologies used for front-end development and include everything from the structure, behaviour, and content seen on the web browser when websites and web applications are opened up. Other web technologies are also used in the front-end development depending on the framework selected, such as Django uses Python and React, Angular uses JavaScript. Earlier, as the client-side had comparatively simpler tasks than now, HTML, CSS and JavaScript were enough

to support the development. Lately, new frameworks have been developed one after the other to support complex web development such as React, Angular, Vue and more.

Back-end

Back-end or server-side web development deals with the core functioning of the web application and makes a website more dynamic. It involves storing and organising the data, ensuring everything in the front-end works fine. It is the part of the web application hidden from the user and cannot be interacted with by the user. However, the functions and characteristics of the back-end, such as calculations, database interactions, application logic etc., are accessed indirectly using the front-end. The programming languages used to develop the back-end includes PHP, C++, C#, Java, Python, JavaScript, and the important frameworks include Express, Django, Spring, and more.

Distributed Systems

With the increase of computational load on computing infrastructure by high-performance web applications, it is not efficient to put the load on a single machine. A single machine might not be able to handle such a large load, and it is better to share the load among a network of machines which increases the availability. According to (van Steen & Tanenbaum, 2016), “A distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system.” The distributed system is composed of several autonomous computers connected by a computer network, and these computers interact with each other to achieve the common goal. Such systems provide scalability as the system can be easily expanded by adding machines and redundancy as the system functioning will not be stopped in case of individual machine failure.

The two primary architectural styles used to implement distributed application architectures are Service-Oriented Architecture (SOA) and Resource-Oriented Architecture (ROA). Distributed systems are composed of components that clients and other components consume over the network through interfaces. These components are referred to as services and resources, respectively, for SOA and ROA. A resource is a distributed component that can be managed through a consistent, standardized interface, whereas a service represents the execution of a requested action. In SOA, several stateless processing units are put together as services. SOA consists of main storage, such as a database, as well as many stateless processing units and an application that connects them. Users send data to these computing units, which

processes it and either return it or change the system's state. The entire system in SOA is made up of services that execute some operations. ROA is built on the concept of resource, which supports four operations, PUT, GET, POST, and DELETE. The operation GET allows the users to retrieve information, and the system state can be changed through PUT, POST, and DELETE. ROA is realized with REST architectural style, which is explained in the following section.

Representational State Transfer (REST)

REST is an architectural style used in designing web services employed in the world wide web as a set of guidelines for creating stateless, reliable web APIs. It makes the communication of the systems easier by making a clear division of application state between client and server. In this architectural style, the client and server implementations are done independently without knowing each other, making it easy to change the client-side or server-side code without affecting its counterpart. It separates the back-end concerns from the user interface part, which improves the flexibility of the application across platforms and scalability. In addition, the systems that follow the REST principles are stateless, which means that the client does not need to know about the server's state and vice versa. The application state is stored only on the client, whereas only resources and their states are stored in the server. Resources can be anything a URL represents, such as an object, document, or information that needs to be stored or sent to other services. The statelessness can improve reliability and scalability as the components can be easily managed and changed without affecting other components or the entire system (Janne Kuuskeri, 2014).

During the development of the prototype in the scope of the thesis, some general patterns based on REST principles were used in the design of the web application. The web application provides access to a queriable and modifiable set of resources, accessed through a unique identifier called URL. Considering the significance of URLs in determining which data must be queried or changed by request, the web framework of choice should provide strong support for defining URL patterns and directing to the appropriate code in the server. These patterns are called a route.

An application can have several request methods for each resource, each performing different actions on the resource. The most commonly used request methods for web applications are GET, POST, PUT, DELETE, where:

- **GET** is used to retrieve a specific object or information
- **POST** is used to create or update the information or elements within a specific resource
- **PUT** is used to replace the information of the entire resource with information received within the request
- **DELETE** is used to delete the resource from the server. How each method should function is defined by the REST architectural style.

The REST architectural style sets the boundary by defining the constraints on how each method should behave (e.g., GET should not update the information in the resource).

The web application should respond using standard formats for each supported combination of resource identifier and request method. When utilising REST over HTTP, this implies encoding the query information in HTML, XML, or JSON and using HTTP status codes (e.g., delivering a 404 error when a resource is not available). The user can also give a list of supported response types, which the server is required to satisfy the requirement (Serrano Mena, 2019).

4.2 Autodesk Forge

Autodesk Forge is a cloud development platform, a Platform as a Service (PaaS) provided by Autodesk. This platform exposes a set of web service APIs that helps implement innovative cloud-powered applications that connect the workflows in the AEC industry with visualisation, engineering, and design. These web service APIs, also known as Autodesk Forge APIs, helps developers in visualisation, data management, model derivatives and design automation and also help to connect with other SaaS products offered such as Autodesk's BIM 360, Fusion 360, Insight 360 etc. In addition, Forge APIs, based on web standards like RESTful APIs, helps with accessing and using design and engineering data, help developers and clients to build and deploy web applications in the AEC industry.

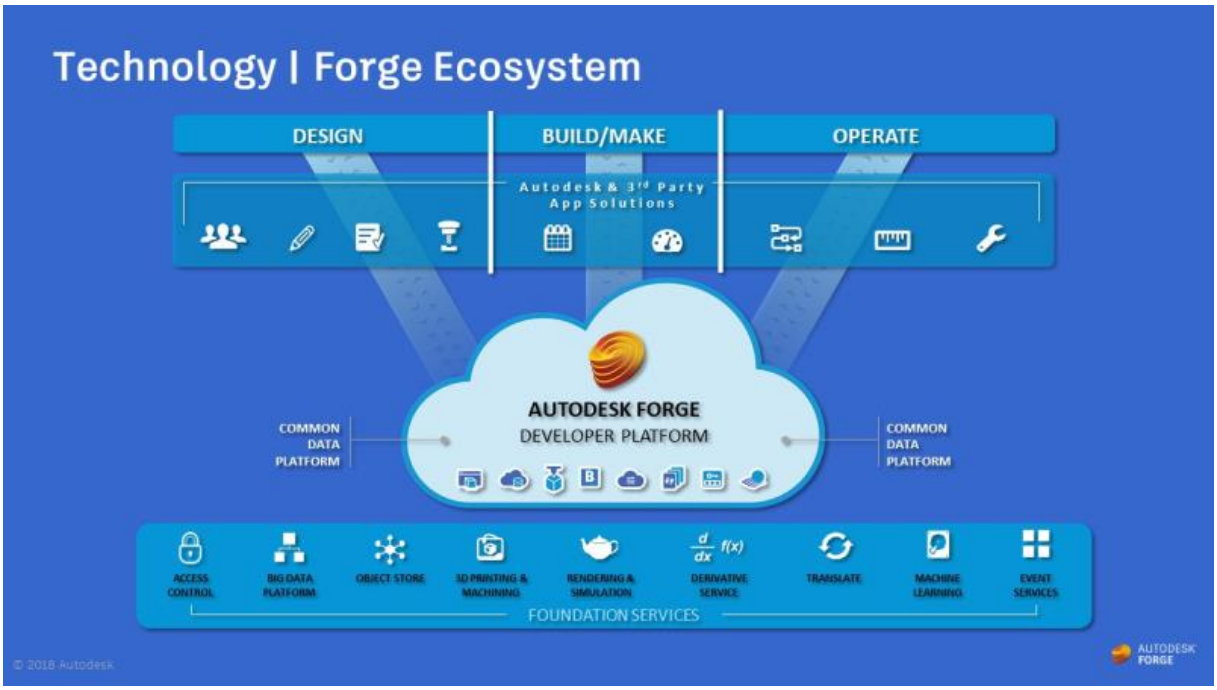

Figure 19: Autodesk Forge ecosystem (Autodesk Forge, 2019b)

4.2.1 Autodesk Forge APIs

The available Autodesk forge APIs as of 01.12.2021 are depicted in Figure 21. Among these, Data Management API, Model Derivative API, Viewer API and Design automation API are the Forge APIs mainly utilised over the course of the thesis for implementing the prototype.

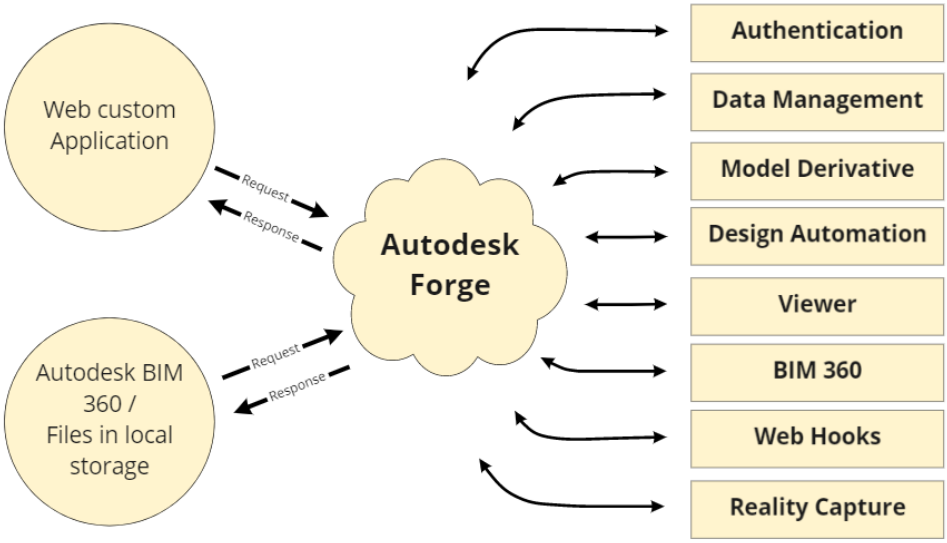

Figure 20: Web application structure using Forge APIs (adapted from Autodesk Forge, 2021a)

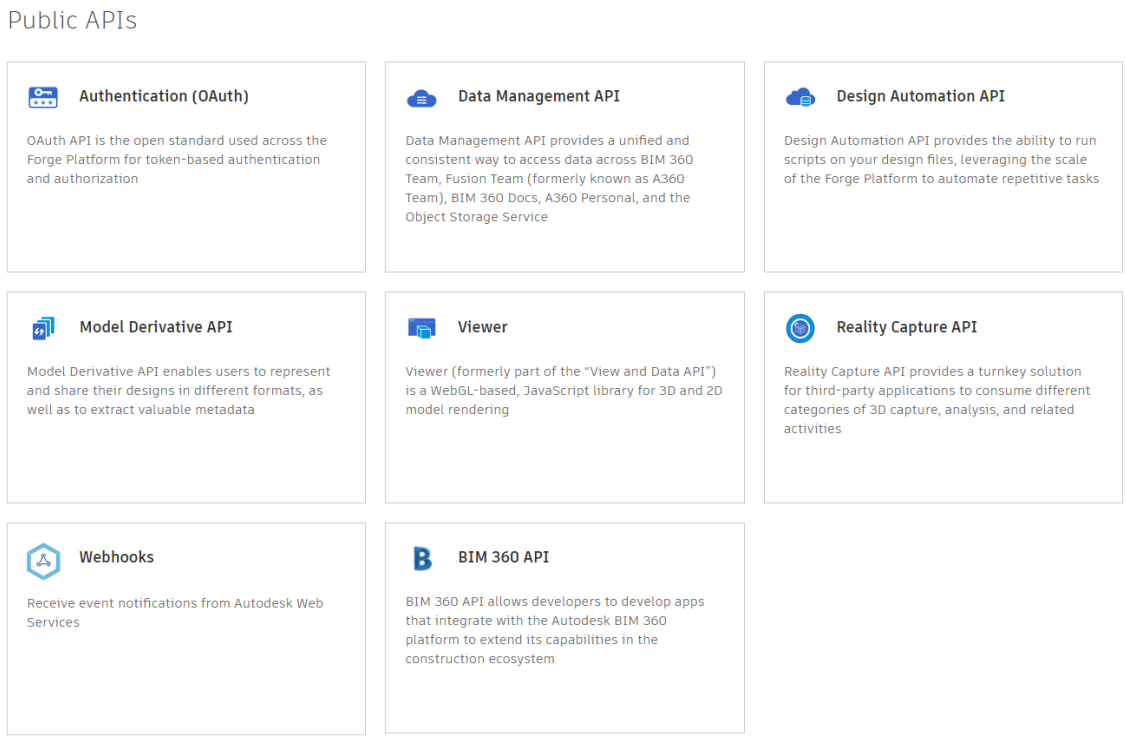

Figure 21: Publicly available Autodesk Forge APIs (Autodesk Forge, 2021b)

Authentication:

OAuth API[7], specifically OAuth2[8], is used for authentication purposes, and there are two types of authentication. First is the 2-legged authentication, where the application (a server, desktop or web app) directly requests an access token from the Autodesk server and invokes the Forge APIs without the need for user interference. This type of authentication is mainly used in Object Storage Service (OSS) API, enabling the application to download and upload files and create a bucket[9] to store the objects – BIM models in the prototype context. It is called 2-legged as it consists of two parties – the user and the third-party application. Second is the 3-legged authentication, where the application explicitly requests consent from the end-user for the use of Forge APIs to access and manage the data. This method redirects the end-user to the Forge

[7] URL: https://forge.autodesk.com/en/docs/oauth/v2/developers_guide/basics/

[8] OAuth2 is an industry standard authorisation protocol that grants a web server's API client limited access to user data. URL: https://oauth.net/2/

[9] Bucket are the spaces created by the applications to store objects for later retrieval.
URL: https://forge.autodesk.com/en/docs/data/v2/reference/http/buckets-POST/

authorisation server for verification. It gets its name as it consists of an additional party - Forge authorisation server in addition to a 2-legged approach. This approach is mainly used to make the application more secure, especially when using BIM 360 to access and manipulate the data. Users can select the type of authentication as either one or mix both approaches, based on the requirement in terms of security and features the application should possess (Authentication (OAuth), 2021).

Data Management API:

This Autodesk Forge API is used to manage the data across BIM 360, Fusion 360, and Object Storage Service (OSS). It includes Project, Data, OSS and Schema services where:

- **Project** services provide access to projects such as a project from BIM 360 Docs.
- **Data** services provide access to metadata such as folders and files in a folder.
- **OSS** service allows an application to download and upload files in buckets. The bucket has a retention policy that determines the amount of time the files can be stored. There are three types of retention policies, Transient policy where files are stored for 24 hours, Temporary policy where files are stored for 30 days, and Persistent policy where files are stored until deleted.
- **Schema** services enable the application to understand the structure of extended data types, such as Fusion designs[10].

In addition, Data Management API uses OAuth2 for authentication and JSON API[11], a strict form of JSON, to exchange information between clients and users. (Data Management API, 2021).

Model Derivative API:

The Model Derivative API extracts metadata – design and engineering data from the model and helps to integrate it with the application. For example, it can obtain geometry, model properties, thumbnails, and even extract part of a design from a

[10] URL: https://forge.autodesk.com/en/docs/data/v2/developers_guide/overview/

[11] URL: https://jsonapi.org/format/

provided model from over 60 different file formats. Furthermore, these Forge APIs translate a file or design to different formats[12] (Model Derivative API, 2021).

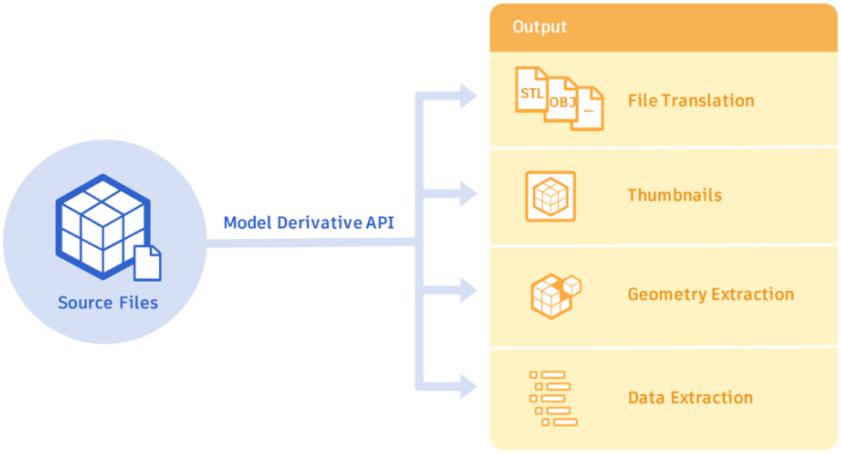

Figure 22: Model Derivative API features (Autodesk Forge, 2021c)

Viewer API:

Viewer API is used to view interactive 3D models directly in the browser without installing additional software. However, before the Viewer API is used to render models in the browser, the model data for the viewer should be prepared using Model Derivative API along with proper authentication. The model data for viewing can be fetched using Model Derivative API from over 60 different file formats, including IFC, RVT, DWG, etc. Viewer API is a WebGL[13]-based JavaScript library that enables the 3D and 2D model data to be rendered in the browser and provides access to the model and its components data without the need for CAD software. The viewer takes in a Simple Vector Format[14] (SVF) file, which can be obtained using Model Derivative API, and then converts to WebGL for the browser to render so that the browser does not

[12] The supported file translations, currently including over 60 different types of file formats, are provided in the following link:
URL: https://forge.autodesk.com/en/docs/model-derivative/v2/developers_guide/supported-translations/

[13] Web Graphics Library (WebGL) is a JavaScript library that enables rendering interactive 2D or 3D graphics in the web browser without additional extensions or plugins.
URL: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API
To test whether the web browser supports WebGL, visit https://get.webgl.org/

[14] Simple Vector Format (SVF) is a vector graphics format that supports hyperlink and layer information
URL: http://www.softsource.com/svf/

need to install any additional plugins. Viewer API is implemented using JavaScript in the front-end, and it is the only API in the Forge ecosystem that deals with the visualisation and front-end related development.

The viewer requires a WebGL-canvas supported browser:

- Chrome 50+
- Firefox 45+
- Opera 37+
- Safari 9+
- Microsoft Edge 20+
- Internet Explorer 11

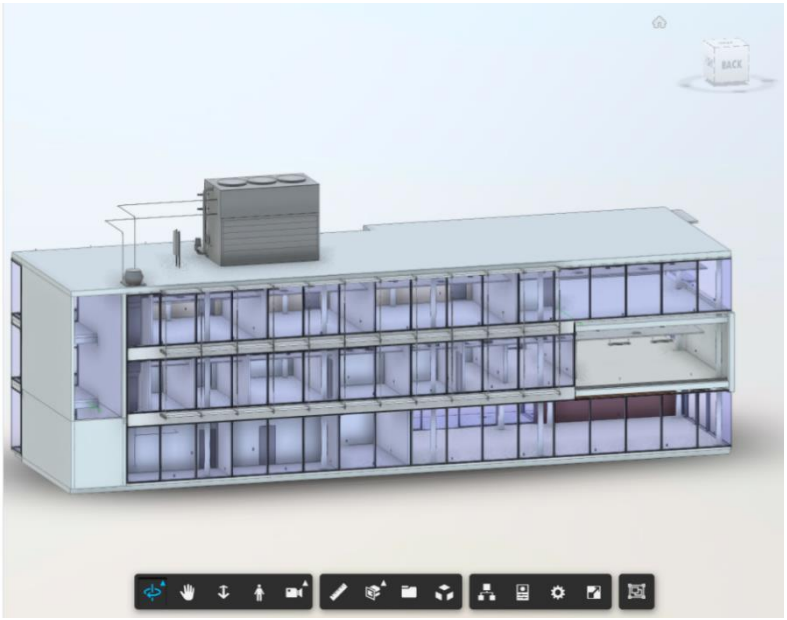

Figure 23: Forge Viewer rendering a BIM model

Design Automation API:

The design automation API provides access to the automation capabilities of Autodesk's core products as cloud services. Currently, it supports AutoCAD, Inventor, Revit and 3ds Max and plans to extend it to other design applications of Autodesk sooner. As a result, most of the tasks that can be automated in the desktop application of the above-mentioned Autodesk products can be automated using this Forge API. Moreover, this web-based API enables to connect these CAD engines running on the cloud with web applications.

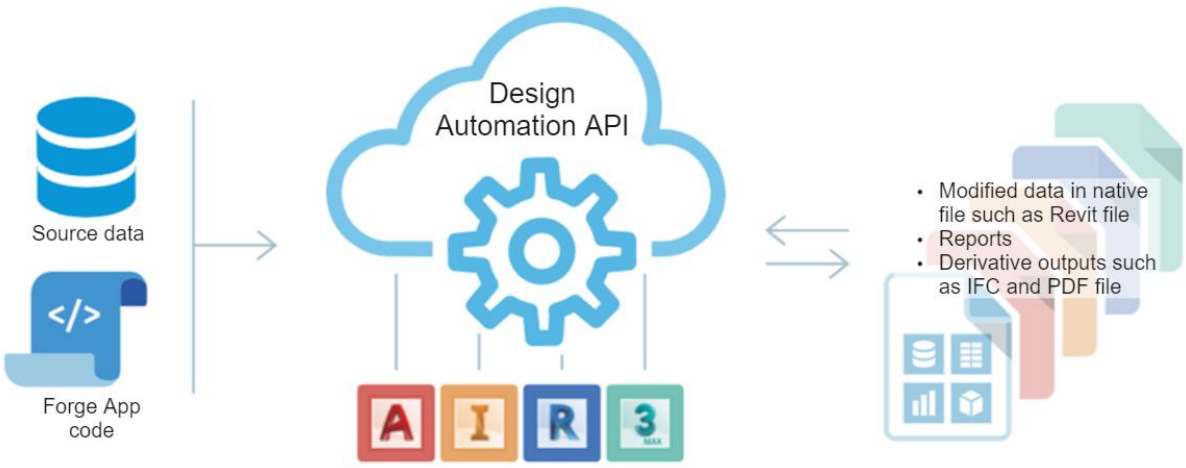

Figure 24: Design automation in Forge (Autodesk Forge, 2020)

Throughout the thesis, design automation API for Revit is employed for prototype development. The software architecture of employing the various Forge APIs in the web application is depicted in Figure *25*. It is basically a Revit's engine running in the cloud as a cloud service. Design automation APIs provide access to the entire Revit DB API functions, enabling access to Revit data and making it possible to build cloud or web apps and services that could create, extract, and modify Revit model data. As a result, workflows such as automating model creation, exploring and analysing model data, extracting reports from the model, and modifying existing models are now possible without the conventional Revit desktop application requirement (Design automation API, 2021).

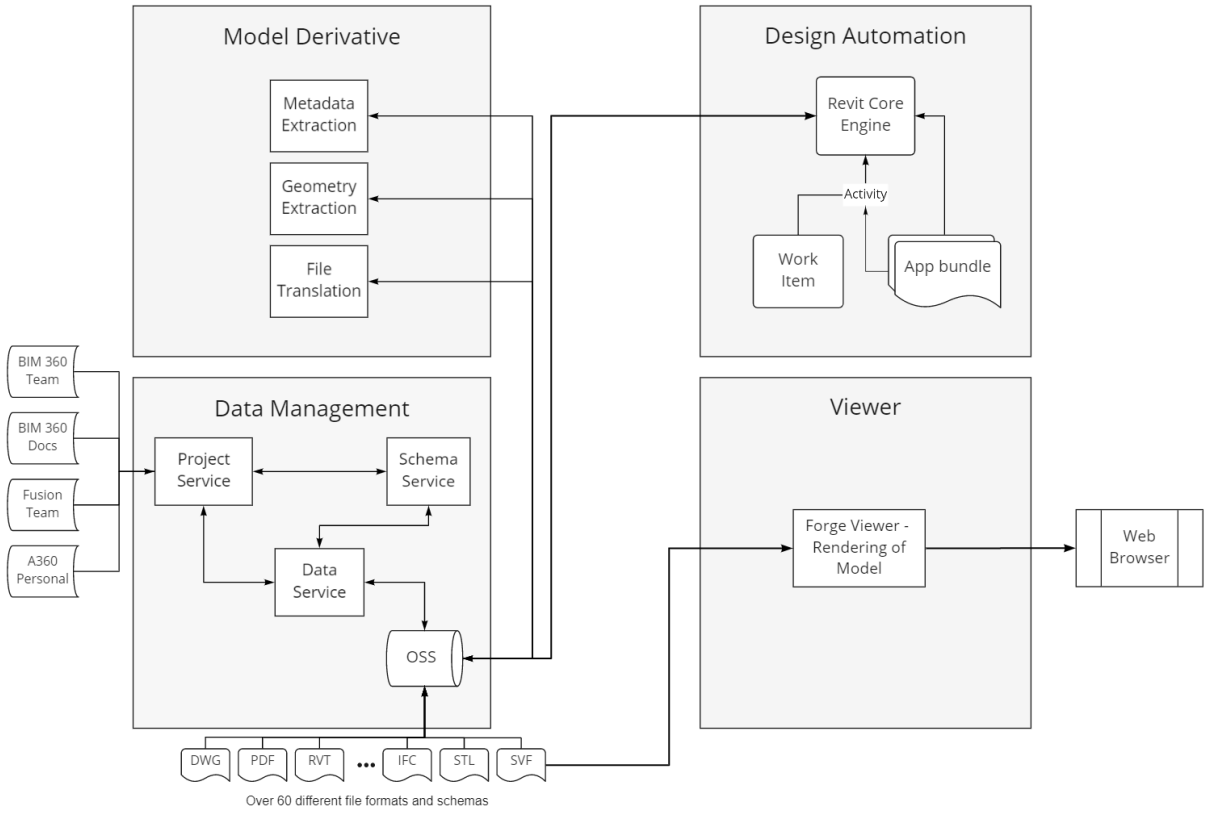

Figure 25: Software architecture using Forge APIs for the prototype (adapted from (Autodesk Forge, 2017))

The use of Autodesk Forge APIs does not come free of cost. Similar to the standard SaaS price model, Autodesk Forge also utilises the pay-as-you-go pricing model such that the costs depend on the use of Forge APIs and services, which are necessary to complete the tasks in the cloud. In order to measure the use of APIs and services offered in Autodesk Forge, a virtual currency called Cloud Credits is used. These cloud credits can be purchased at Autodesk and start at $100 for 100 cloud credits which are valid for 12 months. Many of the Forge APIs and services are free of cost, and some, like design automation APIs and model derivative APIs, need cloud credits to function, as shown in Figure 26.

APIs and services	Costs	Info
BIM 360 APIs	No additional cost	With trial or subscription
Autodesk Construction Cloud APIs	No additional cost	With trial or subscription
Data Management API	No additional cost	With trial or subscription
Viewer	No additional cost	With trial or subscription
Webhooks API	No additional cost	With trial or subscription
Design Automation API	4.0 6.0	Cloud Credits / processing hour (AutoCAD) Cloud Credits / processing hour (non-AutoCAD)*
Model Derivative API	1.5 0.2	Cloud Credits / complex jobs Cloud Credits / simple job
Reality Capture API	3.5	Cloud Credits per gigapixel processed
TokenFlex API**	No additional cost	With trial or subscription

Figure 26: Autodesk Forge APIs and Cloud Credits as per 01.12.2021 (Autodesk Forge, 2019a)

Autodesk offers a 90-day trial where 100 cloud credits are provided. All available APIs and services, along with 5 GB storage and full access to tutorials, training and expert help, can be availed during the trial. In addition, the trial offers an opportunity to experiment with the Forge platform and see whether it fits the requirements and workflows, figuring out which APIs it would need and how many cloud credits would it consume.

4.2.2 Application development using Autodesk Forge

The development of a web application utilising Autodesk Forge APIs, especially design automation, involves several steps in addition to normal web development. In order to get started with the development, it is necessary to create a Forge account, install a preferred IDE (Visual Studio Community 2019 in case of ASP.NET implementation), necessary libraries, and it is recommended to know using Forge APIs. Autodesk provides a developer's guide, documentation and code samples on how to start with the development employing Forge APIs.

Step 1: Create a Forge account

An Autodesk Forge account needs to be created to start with the development. It is necessary to create an account to access the Forge APIs and services in the web application.

Step 2: Create an App in Forge account

An application must be set up in the Forge account where the required Forge APIs are selected and a callback URL is entered. After setting up the application, a client ID and client secret are obtained, which are essential for authentication purposes for the web application.

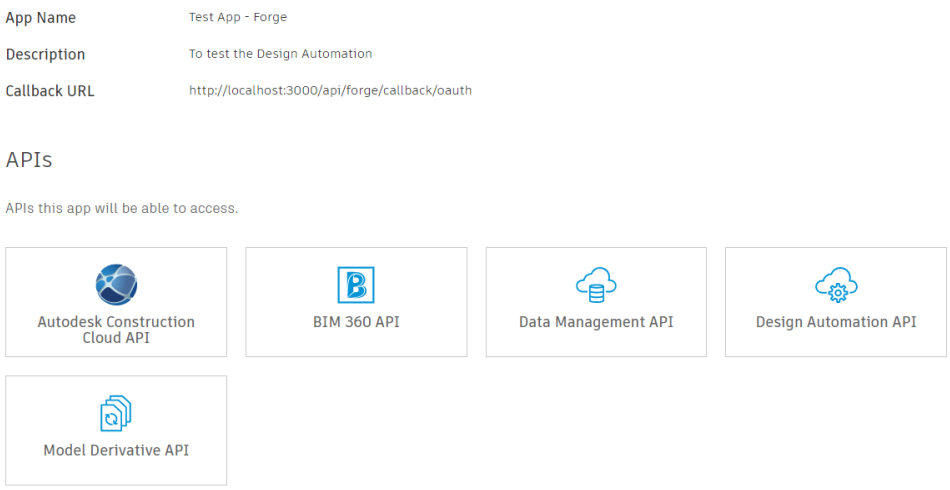

Figure 27: Creating an App in Autodesk Forge

Step 3: Create a server

A local development server can be created using either Node.js or .NET Core. For the purposes of a prototype implementation, .NET Core was selected, and Visual Studio was used as IDE. The Forge client id and client secret from the created app were set as environment variables to obtain an access token while communicating with the Forge platform for authentication.

Step 4: User interface

A user interface for the web application needs to be created as part of the client-side. It is similar to that of any web development (see section 4.1) and can be implemented using HTML, CSS and JavaScript.

Step 5: Upload, translate & show 3D models on the web application

The models can be uploaded from the local storage or loaded and viewed directly from cloud storage services like BIM 360. The models were uploaded from the local storage for prototyping purposes, and they will get uploaded to the Object Storage Service provided by Autodesk Forge. In order to upload, translate and view the

model on the web application, the code can be executed in the local development server and open the local port in the web browser. Figure 28 shows the uploaded project using Forge viewer in the web browser ran locally.

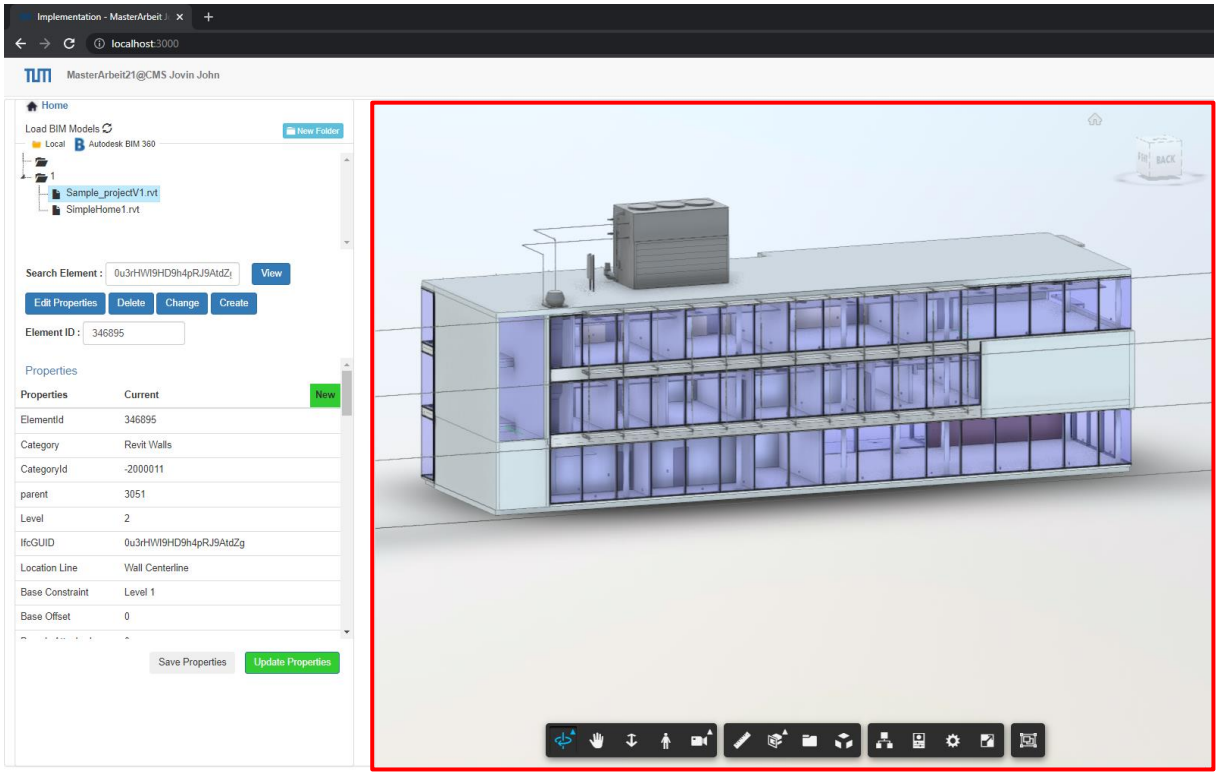

Figure 28: Loading and viewing models in Forge viewer (of the implemented prototype)

Step 6: Prepare design automation capable plugin

A plugin that can perform the design automation tasks needs to be created; the tasks that need to be performed using the CAD engine running on the cloud while running the web application. The respective CAD application is required to be installed to debug and test the plugin locally. If an existing add-in is used in CAD applications locally, it can also be converted to a design automation add-in.

Over the course of the thesis, the Autodesk Revit engine is used for design automation. The plugins created for design automation tasks are similar to conventional plugin development for Revit desktop applications. Finally, a zip file is created, including the binaries and supporting files that make a Revit add-in, and then it needs to be uploaded as an *AppBundle* to design automation.

Step 7: Define and execute an Activity

An *Activity* is an action that can be executed using a specified CAD engine in design automation. *Activities* should be created and published to run specific *AppBundles*. An *Activity* specifies the input files, output files, and the *AppBundle* to use. A job that executes a defined *activity*, using the specified input files and generating the output files after processing in design automation, is called *Workitem*. When a *WorkItem* is posted to design automation, it instructs the design automation to execute the specified *Activity*. For the case of the prototype, Data Management API is used to access the Forge Object Storage Service (OSS) for the storage of models. The Revit file uploaded to the OSS is taken as input for the design automation, and the generated output file is uploaded back to the OSS.

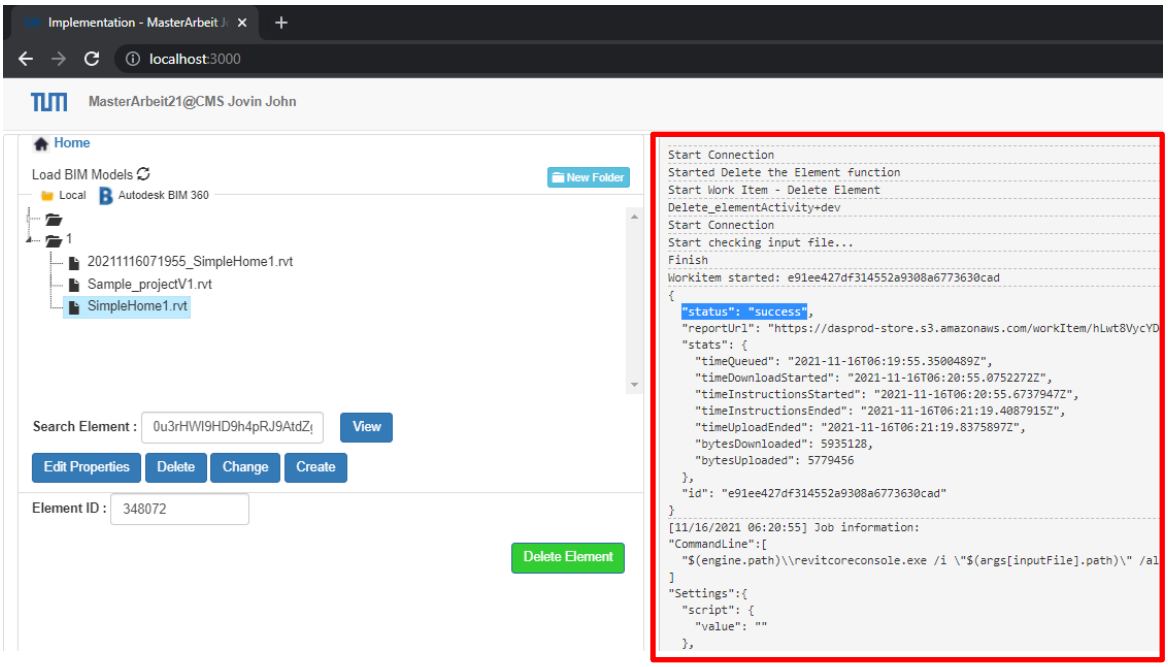

Figure 29: Running a design automation task - executing a *WorkItem*

Figure 29 depicts the process of running a design automation task in the web application. In this case, a *WorkItem* to execute an activity to perform deletion of an element with Element ID as provided in the UI is posted to design automation. The selected file (SimpleHome1.rvt) in the object browser is provided as the input file for design automation. The required parameters (such as Element ID in this case) are embedded as JSON data along with *WorkItem*. After completing the design automation task in the cloud, the log is updated with the task's status, in this

case as ‘success’, and the output file is uploaded back to the OSS and updated in the tree (20211116071955_SimpleHome1.rvt) as shown in the image. Postman[15] is a popular tool that can also be used to test the servers created.

Step 8: Deployment of Web application

The deployment of the web application is the final step where it goes live. There are mainly various possibilities for deployment. Commonly known services include providers like Amazon Web Services (AWS), Microsoft Azure, AppHarbor and Heroku.

[15] URL: https://www.postman.com/

5 Design and Development

This chapter outlines the web application prototype created to test cloud-based design automation. It starts by stating the functionalities in more detail, including the various scenarios. Then, the web application architecture for the proposed prototype and its user interface is further explained, detailing the essential components and concepts used. The evaluation of the prototype is discussed in the following chapter 6.

5.1 Functionalities and use cases

In this section, four functionalities and their possible use cases are introduced, as discussed in section 3.5, to test the capabilities of cloud-based design automation on different levels of complexity.

1. Updating the properties of an element

When there are some changes to the element, the element needs to be updated, which can be done using the application. During design coordination, if some information of an element is missing, it can be added to the elements without the need for BIM authoring tools. For example, in the case of facility management, suppose a room ventilator has been repaired, and as a result, the airflow rate changes to a new value. In order to update the new airflow rate to the existing BIM model, the proposed web application can be used to change the parameters in the BIM model without opening a BIM authoring tool. This updated BIM model with up-to-date airflow values becomes handy during the Flow or CFD simulations to determine indoor airflow distribution, which is important to determine as it will affect the productivity of the occupants.

2. Deleting elements

Usually, element operations like deletion, creation are done using BIM authoring tools. Instead, when a case arises where a specific item needs to be removed, it can be done easily using the proposed application. If an element seems additional during the model-based design coordination, it can be deleted using this web tool. For example, suppose an unwanted table has been identified in the BIM model; it can be removed from the BIM model using a single click in the web application. Another case could be the deletion of unwanted openings in the BIM model. After coordination meetings, project participants change the position of building services elements such as pipes. In this

scenario, the openings in the previous positions that are surplus can be cleaned using this functionality. Furthermore, the deletion of all elements of the selected element type in the whole model or at a level can also be achieved using the web tool.

3. Updating elements

This functionality can be used for model refining purposes, such as refining from generic walls to specific wall types. Furthermore, compared to deleting or creating elements, replacing elements is highly occurring and significant in the FM workflows. For example, an item is usually replaced with a new one when it is defective or beyond repairable. Similarly, an item is also replaced with a new one while upgrading an item. Suppose, if the existing illumination levels in a room are found to be deficient, one solution would be to replace the existing light with brighter ones. This replacement can also be achieved using the web application. It is also possible with the prototype to update all the elements of the same element type with a new type in the whole model or at a particular level.

4. Inserting a new element

When there is a requirement to place a new element in the BIM model, for instance, setting up new equipment, ‘Insert element’ functionality in the web application can be used. This is the trickiest of all the functionalities being tested, and without access to the UI elements of the CAD engine in the cloud using the Autodesk Forge ecosystem, it is challenging to implement this functionality to the full extent.

5.2 Web application prototype

The development of the proposed prototype includes two parts. First is the development of design automation capable add-ins or plugins locally capable of performing the required tasks, and the next is with the web application development. Design automation capable add-ins are similar to Revit plugins created but without access to the UI components to the Revit interface. The development of design automation capable add-ins is described in Annex A. In this section, the prototype development is explained along with the architectures.

Web application architecture

The architecture of the prototypical web application consists of 3 main components: client-side (front-end), server-side (back-end) and Autodesk Forge.

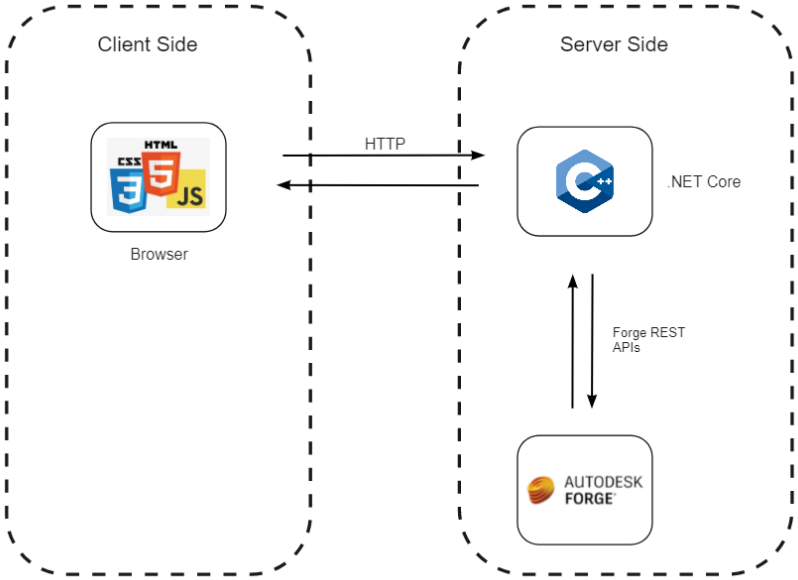

Figure 30: Web application architecture

Figure 30 depicts the basic architecture used to prototype the web application. As stated before in section 4.1.3, web applications have mainly two parts: client-side, which is popularly known as front-end and server-side, which is popularly known as back-end. They both work in the same manner, and the user interacts with the front-end in the browser, whereas the back-end is on the server-side that the user cannot access. In the prototype, the client-side is written on JavaScript, CSS and HTML, whereas the server-side is written using C# programming language in ASP.NET Core framework. The HTTP endpoints were defined in the solution stack with the help of Autodesk Forge code samples to interact with the Revit engine running on the cloud within the Autodesk Forge platform. Table 5-1 lists the implemented endpoints for the prototype web application. While defining HTTP endpoints, the RESTful standard was followed, which is an architectural style in designing web services, as mentioned in section 4.1.3

Table 5-1: Endpoints provided by the API of the developed web service

/api/forge/oss/buckets	GET
/api/forge/oss/buckets	POST
/api/forge/oss/objects	GET
/api/forge/oss/objects	POST
/api/forge/oauth/token	GET
/api/forge/modelderivative/jobs	POST
/api/appbundles	GET
/api/forge/designautomation/engines	GET
/api/forge/designautomation/appbundles	POST
/api/forge/designautomation/activities	POST
/api/forge/designautomation/activities	GET
/api/forge/designautomation/startworkitem	POST
/api/forge/callback/designautomation	POST
/api/forge/designautomation/account	DELETE

The prototype web application that can perform design automation tasks was developed by following the steps described in section 4.2.2. The workflow adopted in the development of the web application is depicted in Figure 31.

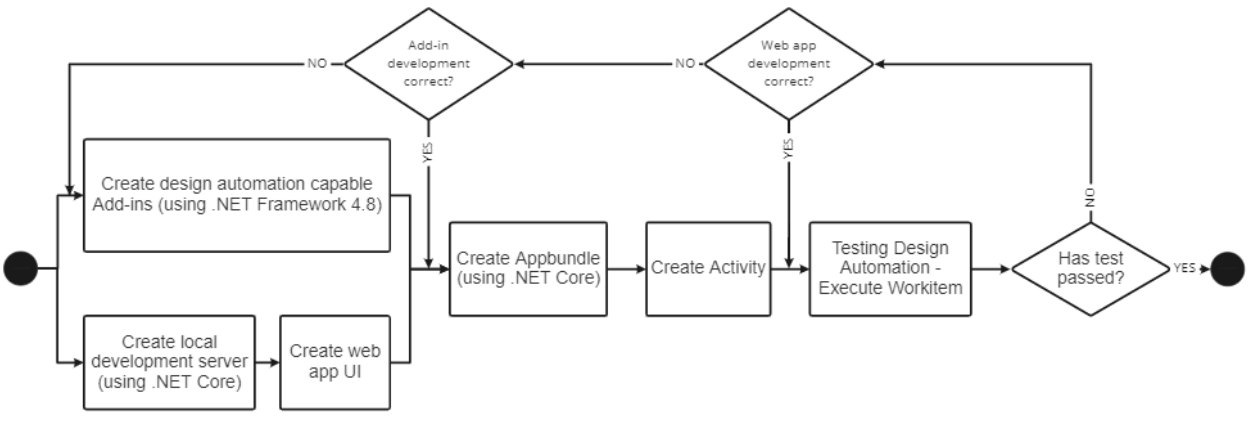

Figure 31: Web application development workflow

The testing of design automation was conducted at each stage to check whether the requirements have been met at performing the design automation tasks successfully. The web application consists of an Object browser implemented using the jQuery[16] JavaScript library. This object browser is used to upload the input file to the OSS bucket in Autodesk Forge. The input files used for the testing comprise .rvt files on which the design automation is applied. The input data required for the design automation is provided from the web application. The input data includes the arguments required by the defined *Activity* in the design automation workflow, such as new parameter values for the selected element in the BIM model. The *WorkItem* is executed by clicking on the button designated for performing design automation tasks. While executing the *WorkItem,* it instructs the design automation to execute the specified *Activity* already published to the design automation. After the design automation task is completed in the cloud, the output file is uploaded to Object Storage Service, which the web application can access. The overall testing process is depicted in Figure 32.

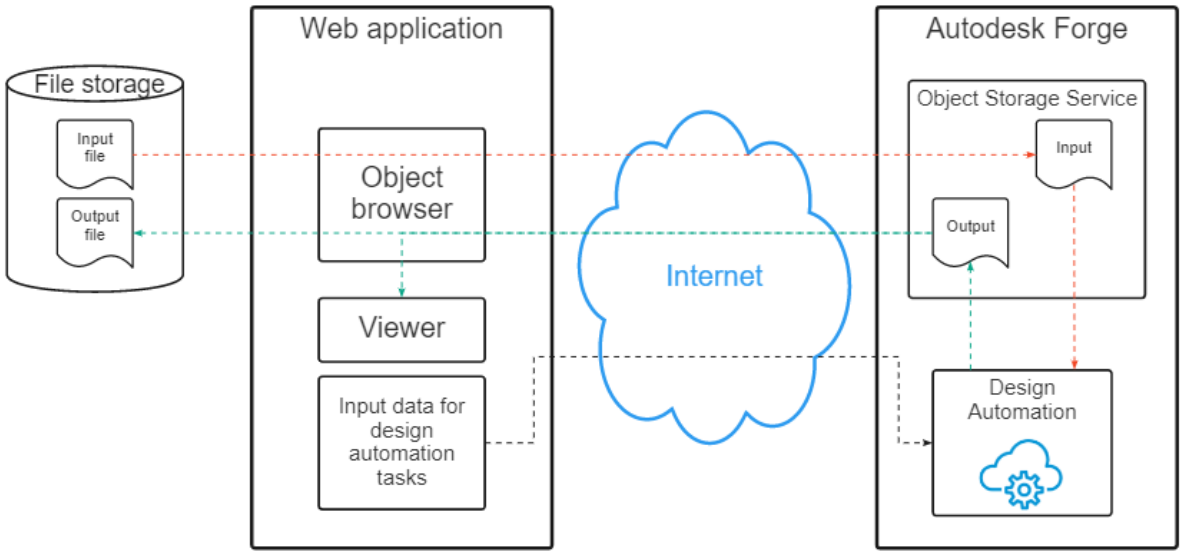

Figure 32: Testing web application

[16] URL: https://jquery.com/

5.3 User interface

In order to create a user-friendly web application, user interface and user experience (UI/UX) design was also considered part of the prototype development. As it is a small scale prototype for exploring the varying complexities of design automation and due to a limited time frame, UI/UX design was done from the inspiration of existing BIM authoring application layouts, as shown in Figure 33. The design of the project browser, content with the model viewer and the sidebar consisting of functionalities provided are inspired from the popular BIM authoring tool Autodesk Revit. In addition, the sidebar for viewing the Activity log was created for knowing the status of the design automation tasks and the working of the web application for testing purposes. This sidebar was inspired by the Output window, which displays status messages in popular IDE, Microsoft Visual studio. The outline of the development of UI is provided in Annex A.

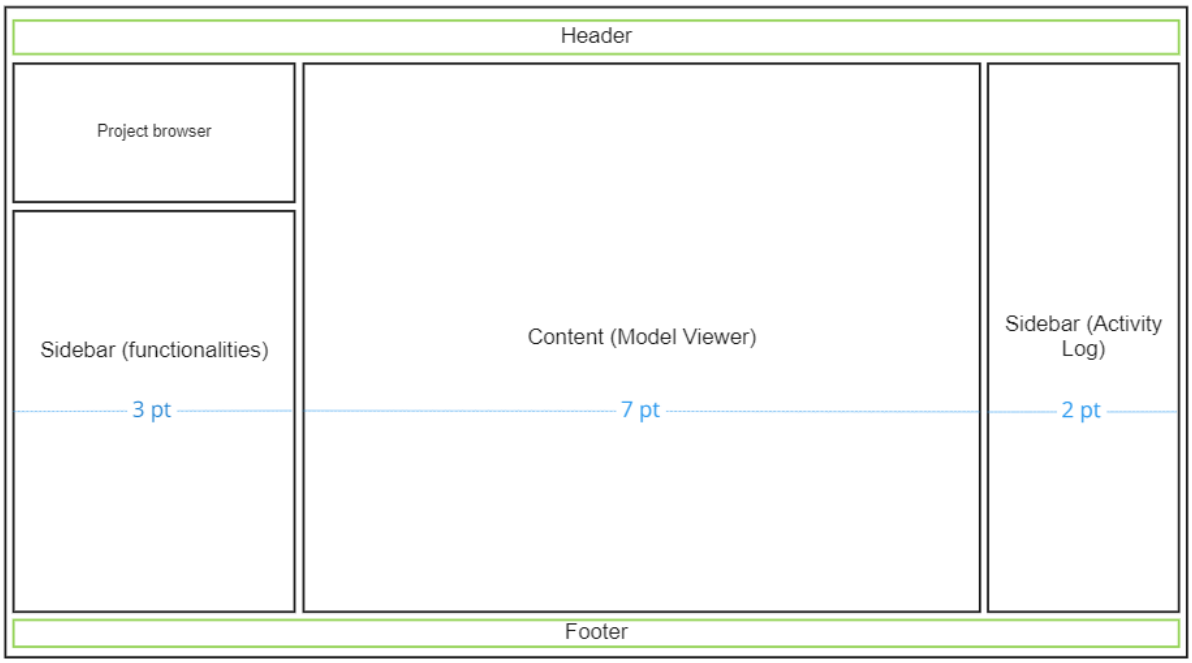

Figure 33: UI design

The user interface was created using vanilla HTML5, CSS & JavaScript (see section 4.1.1). The user interface is divided into five sections as shown in Figure 34: Navigation bar (Panel 1), Object browser (Panel 2), Functionality Sidebar (Panel 3), Forge Viewer (Panel 4), and Activity Log / History log (Panel 5). The UI was created by implementing the principles presented in section 4.1.3.

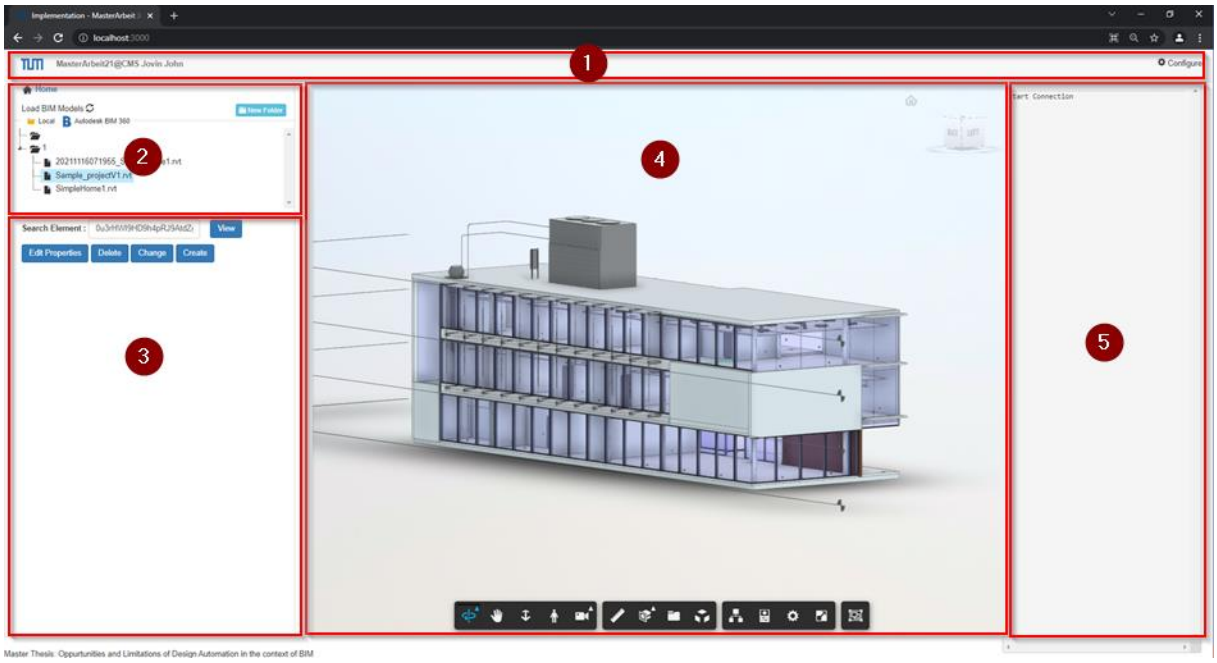

Figure 34: The user interface of the implemented prototype

Navigation Bar:

A navigation bar is an essential component in web development and is displayed on top of the webpage. It usually contains the links to other sections of the webpage and can hold essential information in them. However, in the prototype development, only a slight focus has been given to this part and thereby limited to just providing a logo with a short description along with a Configure button to define the *Activity* by selecting the *AppBundle* and CAD engine in case of any changes to the current *Activity* (see section 4.2.2).

Object browser:

The object browser allows the user to upload the building models and navigate between the uploaded models. In addition, it allows the creation of the folder tree structure and the returned files after the successful running of the design automation task are automatically uploaded to the object browser. Finally, clicking on the model in the object browser allows the model to load and render in the web browser using Forge viewer (Panel 3).

Forge Viewer:

Forge viewer takes up the largest area in the proposed web application and makes it possible to visualise the model. Moreover, it enables navigation in the model and some

functions such as creating sectional planes or viewing properties using the default Forge viewer toolbar.

Functionalities sidebar:

In Panel 2, the following functionalities are included as shown in Figure 35: Updating properties (1), deleting elements (2), changing elements (3) and creating elements (4).

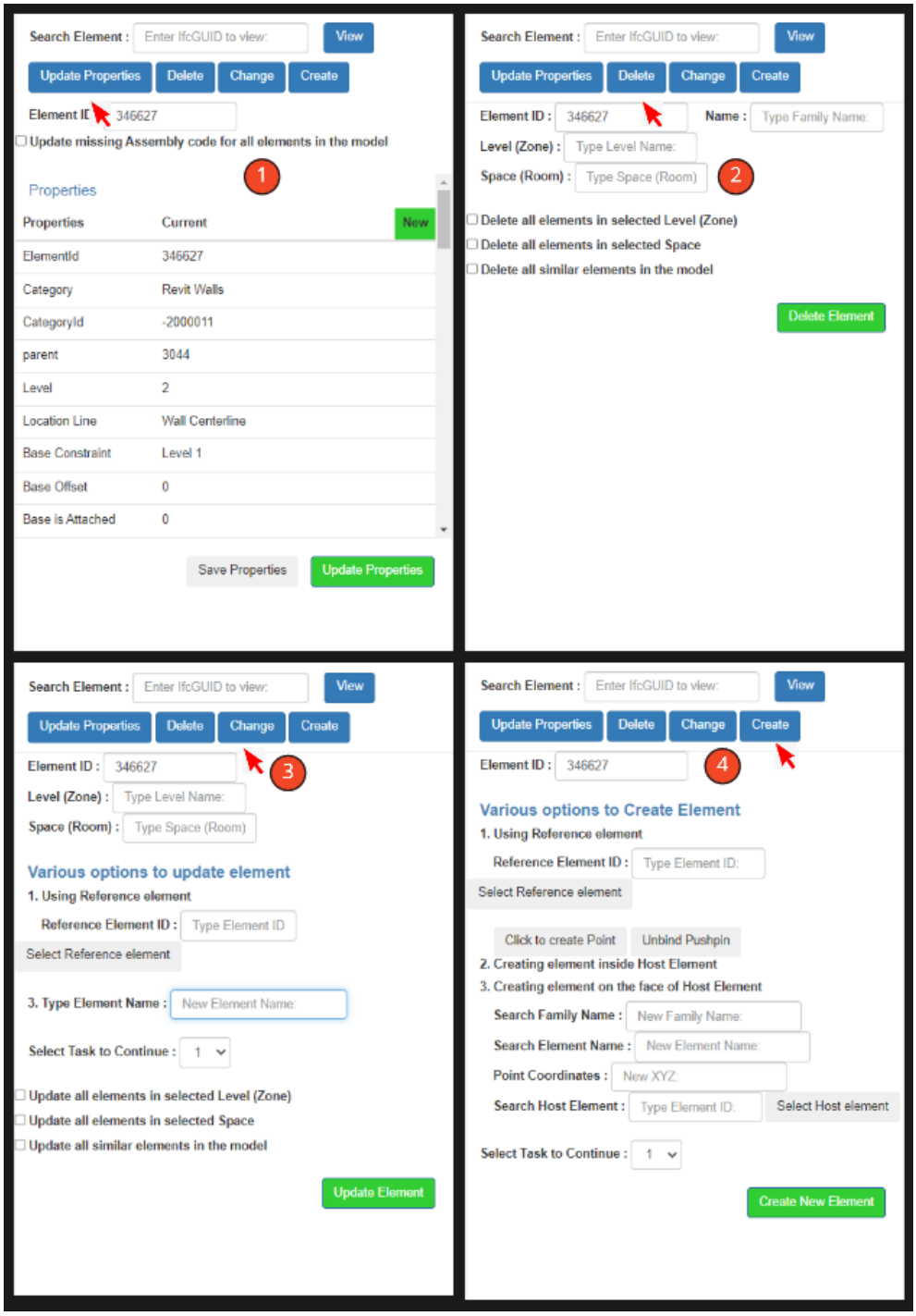

Figure 35: Implemented tabs for required functionalities in the prototype

The sidebar is used to display various functionalities achieved using design automation. It displays essential information from the model using its extracted metadata and provides the option to enter the user input and the button to run the corresponding design automation task. As far as now, four functionalities have been added with relevant information and optionalities. The sidebar could be extended with more information and options to perform the tasks in future iterations.

Activity Log / History Log:

This area displays the information or messages about the application operation and status. These messages are pre-determined and inserted during the development process of the web application, which allows for tracking the activities such as the start and completion of jobs, task status changes, sequential events of the job and so on. The activity log helps to track and control the web application activities and let the user know what happens in the background. This History log is mainly targeted for the development and testing phase of the web application.

6 Evaluation

This chapter includes an evaluation of the opportunities and current limitations of cloud-based design automation. The evaluation is done using the prototype developed in the context of functionalities such as updating information, deleting, replacing and creating elements in BIM models, as mentioned in section 5.1. The chapter is subdivided into two sections, where the testing of the functionalities using test cases are explained in the first section. In the second section, the findings obtained from the results are discussed.

6.1 Testing functionalities

The functionalities of the prototype were tested using two BIM models depicted in Figure 36. These two BIM models were created using the BIM authoring tool Autodesk Revit 2021. One basic residential BIM model of a house was created to test the design automation tasks, and an advanced model provided by Autodesk was used to test more complex scenarios.

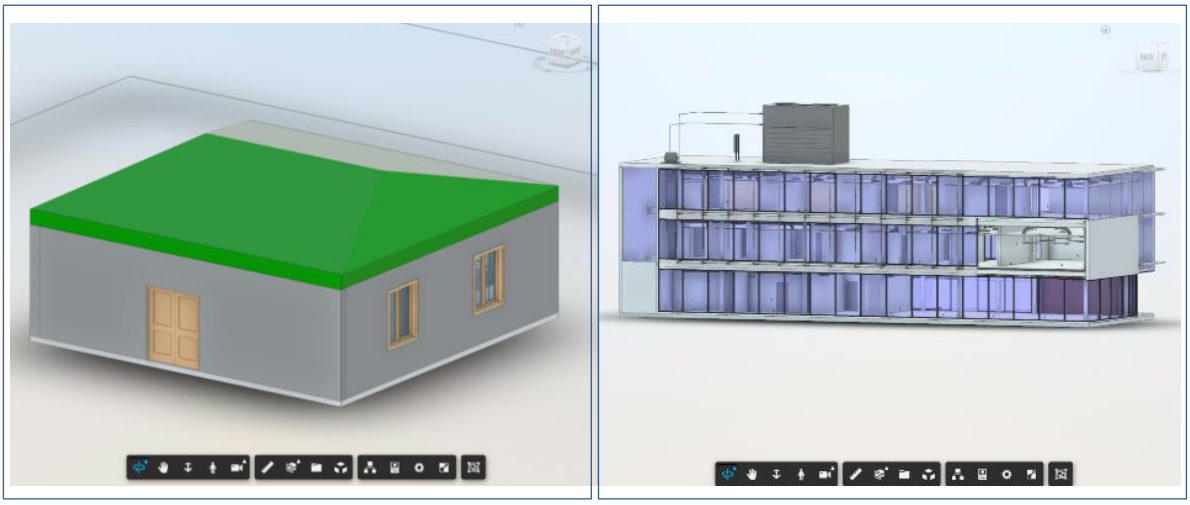

Figure 36: BIM models used for testing purposes

Updating semantic information:

Initially, the model was tested against the case of missing or incorrect semantic information in the form of parameters or properties. The models are constantly subjected to regular model quality checks to ensure sufficient information in the BIM

model. If parameters or properties of an element are found to be missing, the optionality to add such information was implemented in the web tool. The updation of properties is achieved using the ‘update properties’ section, as shown in Figure 37. In the parameters table, the first column describes the parameters, and the second column provides the current value. The new information can be provided in the third column. By clicking on the button ‘update properties’, the parameters could be updated to the selected element in the BIM model. In Figure 37, the missing information of Assembly code and Fire rating in the selected wall has been successfully added using the web application. Even though it was possible to interact with Forge design automation capabilities using the prototype, such data updation in the BIM model using the web application is still manual. It can be regarded as an initial step to the design automation tasks.

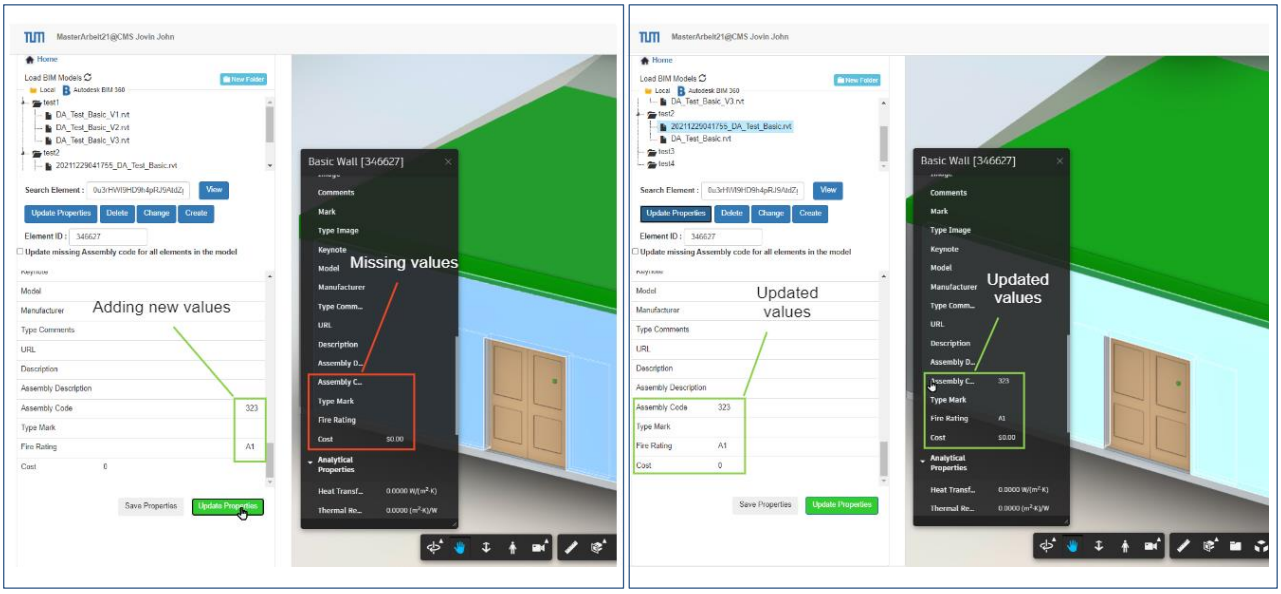

Figure 37: Model element (wall) with updated information (right)

Various possibilities of adding the information were explored while testing. In the web tool, the listed parameters in the sidebar include both family instance and family type parameters in regard to model element family data structure, where instance parameters deal with the parameters directly affecting the particular element in concern and type parameters deals with the parameters common to all instances of the particular family type of the element in concern. While updating type parameters using the web tool, it is found that the parameters of all similar elements in the model (elements with the same family type of the selected element) have been updated as in

on-premise Autodesk Revit software. In this test case, parameters such as manufacturer, assembly code, fire rating, cost, etc. are examples of type parameters. All the elements of the same type have been updated while changing the parameters (in this case – Basic wall: Generic 200 mm). The identity data parameters, analytical properties, dimensions, structural, constraints, approval parameters etc. are found to be able to update using the web tool.

The analysis of various test cases of updating information in a single element provided the possible opportunities of design automation in these aspects. Based on this analysis, a functionality has been implemented and tested that involves design automation where all the elements are checked for specific missing information (in this case – Assembly code). The updation of assembly code in all the elements benefits model-based cost estimation and has been successfully tested using the prototype.

Updating element:

When updating information of an element in case of significant changes becomes insufficient for good design practices, updating elements become necessary in most cases. Suppose the generic wall needs to be refined with a more specific type of wall, such as exterior walls; it can be achieved using this web application as depicted in Figure 38. Furthermore, the functionality to refine the element to a new type for all the elements in the model or at a particular level has been tested. The change of the lamps in the model from a particular type to a new type has been investigated in Figure 39. This feature could be adapted in various use cases when a particular element has to be updated to a new type in various scenarios ranging from design to operations.

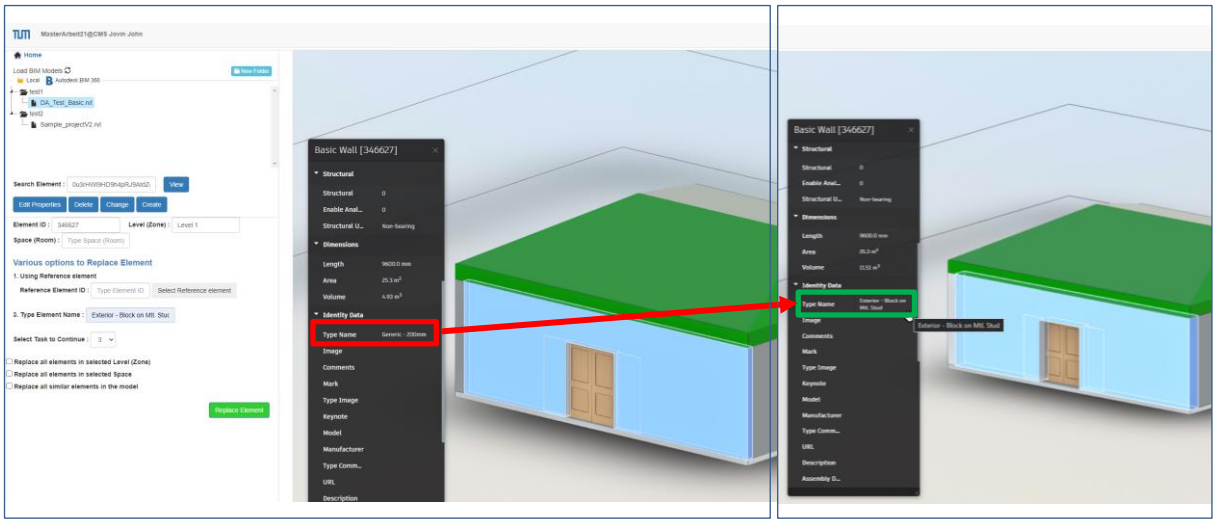

Figure 38: Refining a generic wall (left) with a specific wall type (right)

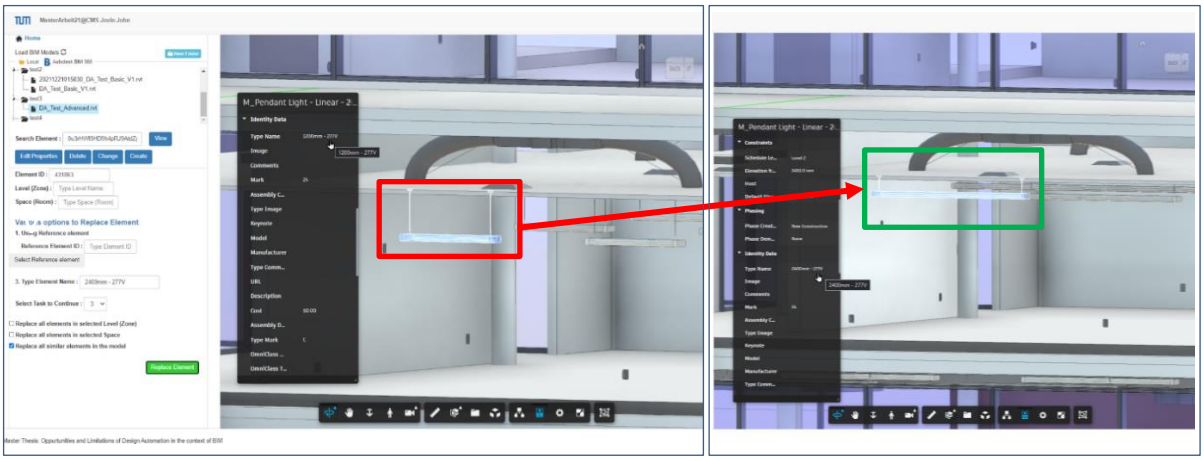

Figure 39: Refining all the pendant lights of type 1200mm - 277V (left) to 2400mm - 277V (right)

Deleting element:

The prototype also provides means of deleting elements from the BIM model. The elements could be deleted either as a single selected element or all the elements of a specific type of the selected element across the whole model or filtered on a certain level. Suppose the function of the room needs to be changed, and instead of separate two rooms, it was decided to make a big single room. In order to do so, the intermediate middle wall is to be deleted, which is done using the web tool, as shown in Figure 40.

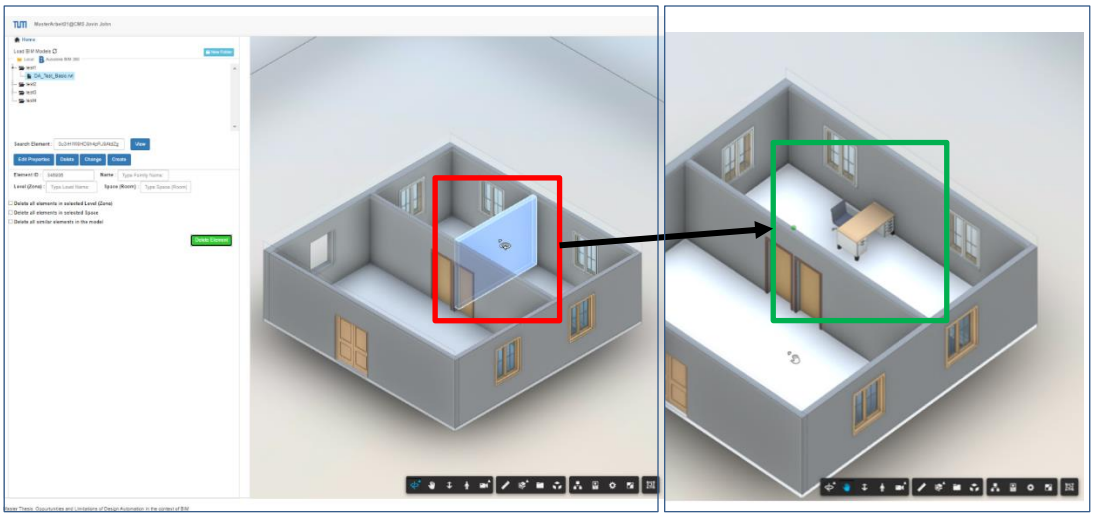

Figure 40: Deleting a selected wall from the model (left) and the updated model (right)

Inserting element:

Finally, placing elements with the aid of this web tool were also tested. When a new element is to be placed and knowing the location in terms of XYZ point and host element, a new element can be placed on the provided host element using the web

application. This could benefit the cases where the elements should be placed on the model at specified location points, such as the placement of columns in a model at provided locations. An example of providing a lamp at the wall is shown in Figure 41.

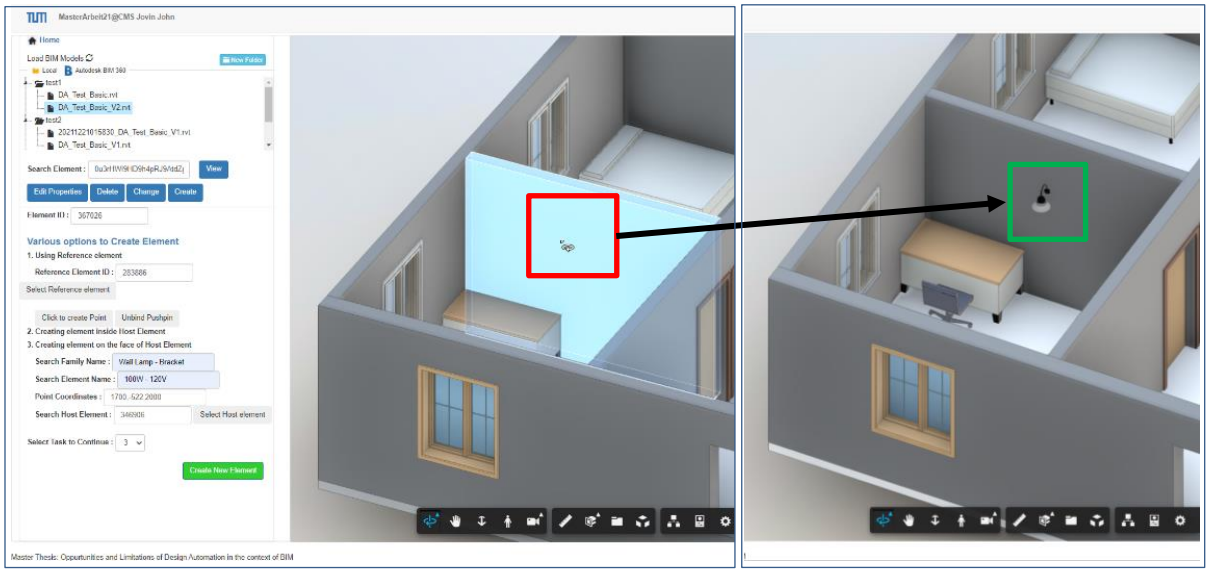

Figure 41: Inserting a new lamp into the host wall in the existing model (left) and updated model (right)

6.2 Findings - Opportunities and Challenges

6.2.1 Updating information

Different scenarios were tested and further explored using the prototype web application. While testing this functionality, various parameters of different elements were tried to update and enrich the model with required or missing data. The prototype can be used to solve the scenarios where missing information needs to be updated in the selected element. During model coordination, when an error has been identified, this web tool could be used to rectify the errors or missing data of a particular element or element type. Even after planning, and construction, the use of BIM is getting more predominant in the FM sector, and while operations, if some parameters need to be updated or changed in the BIM model, this tool could be used for a quick and easy update of BIM information.

Most of the parameters that could be edited using the BIM authoring tools could also be updated using this web tool. The parameters with the storage type as string, int, double are the parameters that could be updated using this tool, which covers many parameters. However, some parameters that can be updated using a desktop-based Autodesk Revit could not be updated in the current development. For example, the

parameter ‘base level’ that specifies the base level of an element could not be updated as it is referenced to another element in the BIM model (in this case, ‘Level’ is an element or family instance). Such parameters that doesn’t have a number or text storage type, cannot be changed in the current development. Also, the parameters that depend on other parameters, such as the parameter ‘area’ or ‘volume’, that depend on other parameters such as ‘length’, ‘width’, and ‘height’, are not updated. These are also not available for editing in the desktop-based Autodesk Revit software. Furthermore, it was not able to update the information in case of analytical parameters such as Heat Transfer Coefficient (U), thermal resistance (R), etc. These are also blocked to update using on-premise Autodesk Revit software, such as in the properties of the ‘Basic wall’ family. However, it is found to be possible to change the parameters that could impact model design changes, such as the ‘height’ of the wall or door, ‘base offset’ etc., but is found to be is challenging to handle these changes. Sound engineering knowledge might be required to deal with parameter information updates that impact such design changes.

Another challenge faced during the development of this functionality is related to unit system. The Autodesk Revit engine in the server of Forge ecosystem development uses feet as the default length unit. The units of the parameters in the forge viewer are set corresponding to the default unit system in the BIM model during the prototype development. Therefore, proper explicit unit conversion of number inputs should be done when dealing with number inputs. Further shortcomings to the prototype include the lack of units of parameter values in the properties tab in the web application. In addition, a grouping of parameters was not achieved in the current state, as seen in the BIM authoring tool’s Properties window. It is a future direction that can be further looked upon.

Even though the prototype was able to update the information of a specific element, there is still manual work in updating such information in the web tool. During the testing, it is understood that the workflow can be adapted to many automation use cases, which needs the information in elements to be updated. One such usecase has been identified and implemented in the prototype where all the elements are checked for missing ‘assembly code’ and updates the assembly code if it is missing. As part of the implementation, the missing values of ‘assembly code’ in basic elements such as walls, doors, windows etc. was identified and updated with their assembly code based on DIN 276. Furthermore, to bring advanced automation capabilities in filling missing

data, a suggestion system can be developed that is able to recognize and suggest the missing information of specific parameter of elements of a particular family or element type. The possibility of the web application to integrate Machine Learning and Artificial Intelligence algorithms could help to extend the automation possibilities apart from using the available Forge APIs in the Forge ecosystem.

6.2.2 Updating elements

Updating elements is a useful functionality and becomes handy in many scenarios. The elements found to be insufficient in the BIM model due to new requirements can be changed to a new type at the exact location. It is common to find the elements that need to be updated with new elements during design phases – using model checks and also during operations phases. When an element needs to be changed to a new type during operations phases, such a platform that could perform easy, quick update of elements is convenient.

Some challenges were identified during the testing and development of this functionality. First, the element type that the selected element needs to be updated with, should be present or pre-loaded to the project before using this prototype. This becomes a disadvantage as all the required families that the user might need to work with in future might not be guaranteed to be included in the BIM model. Pre-loading many element families in advance increases the size and complexity of the BIM model and may not be practical in some cases. Generally, when this case arises while modelling in desktop-based Autodesk Revit, the new family as .rfa file is loaded to the model. This shortcoming to this prototype can be solved by providing the new family as separate .rfa file along with the BIM model as input to the design automation, but it requires further development efforts. Another challenge is refining elements of size other than the original size, which is encountered in Figure 42 and Figure 43. During various tests, it is seen that the new elements are placed in the BIM model at the location where its centre aligns with the element that was being refined, just as in BIM authoring tools. When larger-sized elements are placed, the elements might overlap with other elements in the vicinity. It creates new physical clashes. Resolving such clashes manually as in design authoring tools is difficult in such a scenario due to lack of UI access to CAD engine interfaces in the cloud. Therefore, automating the refinement in such scenarios can be challenging as various factors such as proximity to other elements, element orientation, host element requirement etc., needs to be

considered. Hence, proper care should be provided before updating the elements with different sizes.

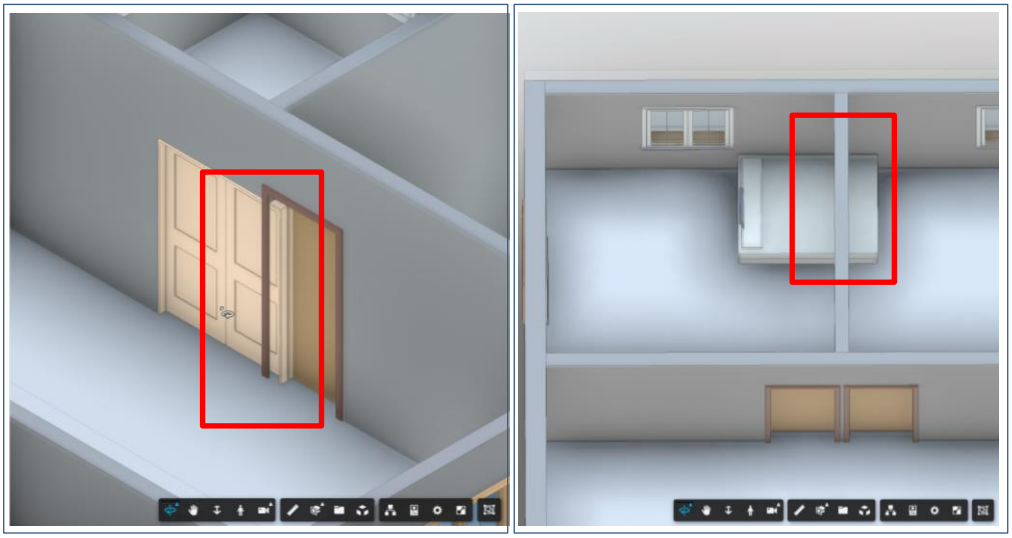

Figure 42: Errors during updation of elements: doors overlaps (left), bed penetrates the wall (right); create clashes

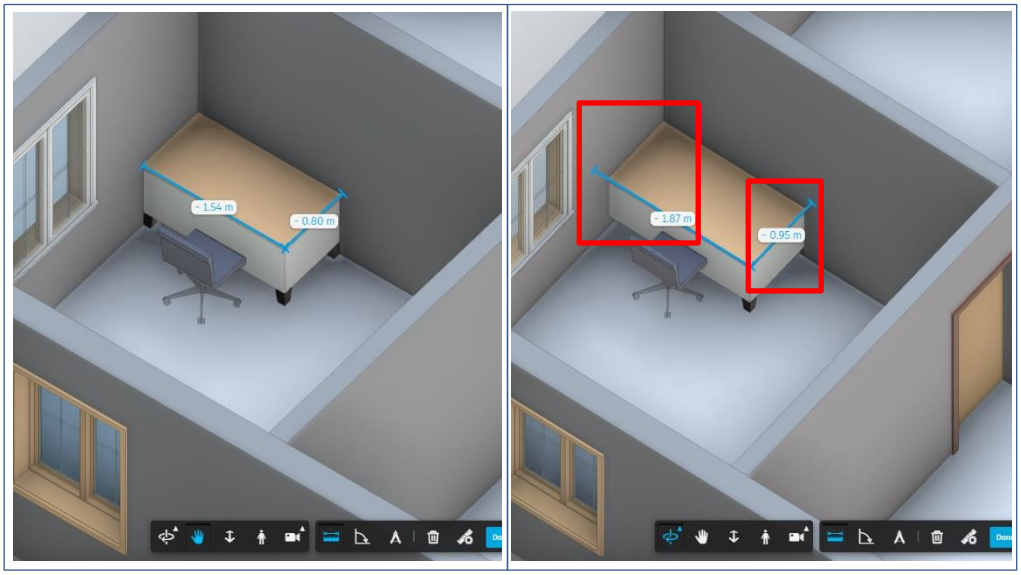

Figure 43: Changing the dimensions (length and width) of an existing table by 'updation' functionality; updated model (right)

6.2.3 Deleting elements

Deleting elements is a useful function that could be employed while updating the BIM model. At times, when some elements are found to be redundant, they could be deleted from the model quickly without any hassle on the web. Various users can use this functionality, including clients, planners, site supervisors, and facility managers, to

delete unwanted elements from the BIM model. Deleting an element is a comparatively easy process among the functionalities tested and involves deleting a single element or deleting the elements with the same element type in the provided level or over the whole model. While deleting an element from the model, the selected element instance and its relationships to the hosted element are also deleted from the model. Suppose an element that hosts other elements such as a wall hosting doors are deleted; the elements hosted by the deleted element are also deleted as in desktop-based authoring tools. Also, selecting more than one element type to be deleted at a time is not implemented with this prototype. However, it is possible to do so with proper adjustments in the UI for consequent user inputs.

6.2.4 Placing new elements

Inserting elements in the BIM model is much more sophisticated in terms of feasibility in the current context of cloud-based design automation. There are various ways to insert a new instance into the model, and it depends on the element type which needs to be placed and the data structure of the BIM authoring tool. In the case of Autodesk Revit, the new elements can be inserted by providing the element type to be placed:

- Into the face of an existing element (host element) in the model (e.g. inserting structural stiffener on wall face)
- Into the model using XYZ coordinate location for elements that do not require host element or level (e.g. adding coordination body[17] to the model)
- Into the host element using XYZ coordinate location (e.g. inserting door into a wall)
- Into the host element using XYZ coordinate location and reference direction (e.g. inserting a bed on the floor with reference direction)

Each case should be implemented separately, and the input~~s~~ for each case differs while performing design automation tasks. For example, the user might have to know the appropriate type of insertion to provide necessary inputs and proceed with inserting elements individually for different types of elements. Automation efforts for detecting each element and its insertion type are challenging but practical for individual use cases.

[17] Original – Koordinationskörper in German

Another main limitation is the lack of UI aspects while using the cloud-based design automation system to place new elements in the model. Conventional design authoring tools presents a UI in which the elements can be placed manually either in 2D or 3D, which is not yet possible using the implementation approach chosen. In this case, the placement of new elements is being fulfilled by explicitly providing the element coordinates, which is complex and not user-friendly. Moreover, the accurate XYZ coordinates of the new element cannot always be assured and are not recommended over manual design practices to place the elements in such a manner as it brings more confusion and uncertainty. Also, during the placement of elements in the model, such as a lamp in the host wall, as shown in Figure 41, the coordinates are being adjusted by the design authoring engine (here - Autodesk Revit) to incorporate the element to fit the hosting element better. Therefore, creating new elements in an existing BIM model using this tool is challenging as far as now due to the unavailability of UI as in design authoring tools, and manual modelling is suggested for such practices that require more accuracy. The possibilities and shortcomings of the developed prototype as described in this section is summarised in the Table 6-1.

Table 6-1: Possibilities and shortcomings of the implemented prototype

Tested Functionalities	**Possibilities**	**Shortcomings**
Updating information	• Can add missing/incorrect information (model enrichment) of the selected element in the model. • Can update both instance and type parameters. • Parameters influencing design changes, such as changes in length, can be updated. • Can identify the missing information of a specific parameter (Assembly code) of all the elements in the model and update the correct value for basic elements such as walls, doors, windows etc.	• Can only update the parameters with number or text input format. The parameters which refer to another element cannot be updated in the current state of the prototype. • Explicit units conversion is required for the number inputs to the imperial unit system (default Autodesk Revit unit system).

Updating elements	• Can refine a generic element with a specific one or an outdated element with a new one. • Can refine all the elements in the model or a level with the same element type of selected element	• The element type to be which the selected element needs to be replaced should be present or pre-loaded in the BIM model. • The size of the new element should be considered, and proper care should be given as the replacement is done with regard to the centre of the selected element.
Deleting elements	• Can delete the selected element or all the instances of the selected element in a model or a level.	• Deleting multiple element types at the same time is not possible at the implemented prototype
Placing new elements	• Places a new element at provided XYZ location and host element	• Lack of interactive UI makes it difficult to place a new element at the required location, as in BIM authoring tools. • Cannot assure element to be placed in the provided XYZ location as the Revit engine tends to alter the location to match the host element slightly. • Difficulty in inserting various types of elements as various elements are inserted generally based on some conditions.

7 Summary and Outlook

7.1 Summary

This study aimed to research the opportunities and the current limitations of cloud-based design automation in the context of BIM.

Most of the BIM authoring software's are standalone and on-premise in nature, which comes up with limitations such as high-end hardware and computing requirements, time and cost, platform-dependent nature, convenience and scalability. On the other hand, cloud-BIM integration can solve such existing problems in the BIM model development. Besides using CDEs to ease the storage and collaboration of BIM models and documents, cloud computing benefits BIM with its high-performance computing capabilities by outsourcing computationally intensive tasks to the server.

The evolution from CAD to object-oriented parametric BIM models helps automate various workflows and laborious tasks. In this context, the opportunity of having the design automation tasks in a location-independent, high computing platform will expand its potential and opportunities. In order to test the opportunities and current limitations of extending the BIM related tasks to the cloud platform, a web application was developed as a SaaS application using the Autodesk Forge cloud development platform. The implementation workflow adopted for outsourcing the design automation tasks to the cloud using the selected implementation approach was validated using the prototype. The basic functionalities (CRUD) for typical design automation workflows include updating information, deleting, updating, and inserting new elements. All these functionalities were tested in a cloud-based design automation system using the implemented prototype.

During the development and evaluation of the prototype, the essential steps that are necessary to implement a cloud-based design automation application were identified. They are:

- justification of the task that needs to be automated and gather sufficient information for the successful development of the web application
- identification of the platform where the cloud-based design automation can be performed effectively

- creation of the implementation workflow or formal model for the identified task that needs to be automated
- implementation using the provided APIs according to the implementation workflow
- evaluation of the prototype for the intended automation tasks
- improvement of the prototype by analysing results and gathering insights from intended users

The testing of the provided functionalities in the prototype helped to understand the opportunities and limitations of such cloud-based design automation systems. The high computing capabilities of cloud computing helps to ease the burden of performing time-consuming and iterative tasks. The time saved by the automation can be invested in more important tasks, which reduces the project duration and costs. The on-premise design automation is usually performed on the BIM authoring tool, such as Dynamo scripts are run on top of Autodesk Revit. In case of large automation tasks, the BIM authoring tool cannot be used while the design automation runs. Outsourcing such tasks to the cloud could also free up the BIM authoring tool, which can be utilised for important design tasks. Furthermore, if the design automated tasks that have been outsourced to the cloud are commonly occurring, the web application which can perform this cloud-based design automation can be used by multiple users in parallel or even scaled to the required demand. This could amount to significant savings in terms of time and money. The opportunity to integrate other technologies could extend the possibilities of cloud-based design automation.

Despite having limitations, such a universally accessible, flexible and scalable cloud-based system with high customisation possibilities to specific needs will benefit the stakeholders, especially those limited with the necessary hardware and software requirements. Therefore, moving the tedious and repetitive tasks during the BIM development to the cloud is highly recommended. Especially as one of the most important priority in the present AEC industry scenario is meeting the deadlines of fast-paced BIM projects cost-effectively.

7.2 Existing challenges

Even though the benefits provided by a cloud-based design automation system are valuable, some challenges and limitations were also encountered during the literature review and the development of the prototypical web application. The main limitation

faced was the unavailability of interactive UI similar to the UI in traditional BIM authoring tools. The Autodesk Forge cloud developer's platform, which was employed for the prototype implementation, does not offer such UI as of now. This limits the live interaction with the BIM model and thereby the functionalities of the proposed system in accessing the UI aspects. It was able to solve this limitation to an extent by incorporating Forge Viewer into the prototype and thereby utilizing its 3D viewing capabilities along with the model data extraction capabilities of the Forge system. Initially, the model was loaded, and then the required information was accessed and made available using APIs. After that, the tasks for design automation were set and then updated the BIM model using the web application. The updated BIM model is then viewed for analysing the results, and these steps could have been reduced if an interactive UI was available as in BIM authoring tools.

Another challenge encountered was the data transfer, which tends to be slower if the input and output files are stored outside the Forge system. The upload and download phase can be optimised by using Forge storage services such as BIM 360 Docs or by caching the necessary files in the object storage service provided by Autodesk Forge prior to running design automation tasks.

Cloud-based design automation can also be affected by latency issues and poor broadband connection. As the upload and download of models along with the rest of API calls to the server are primarily over the internet, good internet connectivity should be ensured for the proper working of such cloud-based design automation systems. The use of distributed cloud architecture can help the latency to be minimum for the specific web application. Proper care should be given in identifying and rectifying the latency issues at the earliest for the smooth performance of such cloud-based design automation. Further improvement in Information and Communication Technologies (ICT) like the emergence of the 5G network can solve the speed and latency issues and accessibility in rural areas without a broadband connection, such as distant construction sites.

One of the important factors that can affect the industry-wide implementation of cloud-based design automation is the industrial partners' reluctance to share the BIM models in the cloud. However, due to the advanced security measures undertaken by the popular cloud service providers such as consistent security updates, built-in firewalls,

backed up data, and end-to-end encryption, the models shared during the cloud-based design automation is considered safe similar to the data stored in the local computer.

The cost of using cloud-based design automation is another factor that has to be taken into account while developing such systems. The design automation costs can be higher than other cloud services due to its requirement of high computing resources to perform automation tasks. Even though the costs are computed generally in the pay-as-yo-go model, a cost analysis of the long-term use of such systems might be necessary before moving into the cloud. With the improvements in ICT technologies, the costs of using cloud computing services are expected to reduce in the future.

Another challenge is the level of programming knowledge required for the implementation and maintenance of the web application directly by the end-user. The end-user, like designers, may not have sufficient programming skills for the development as it is not typically part of their work. The web application has to be created and modified similar to the proposed prototype to fit the use case, requiring advanced programming skills. For efficient implementation and maintenance, it is recommended to have a collaborative approach between the design team or the user who benefits from the web application and the developer‘s team.

7.3 Outlook

Further research and improvement on the implementation workflow must be made to adapt the system to specific use cases. From the evaluation of the prototype, some limitations and challenges were identified on which future research will be beneficial. The implemented web application could benefit from the integration to a CDE such as Autodesk BIM 360 for accessing the BIM models directly. Placing a new element in the existing BIM model using the applied implementation approach seems impractical due to the unavailability of UI. The prospects of having an interactive UI in the web will extend the design automation tasks to broader scope in future. As the AEC industry is focusing more on the cloud, such an interactive UI in the web can be expected soon as it is already found in tools such as Trimble’s SketchUp, Autodesk’s AutoCAD Web etc. The Autodesk Forge is relatively new, and with the improvement in the offered APIs, the functionalities could be extended further, which now is limited to database-level data. Currently, the design automation tasks using the Autodesk Forge platform are done using file-based updates, and each time the BIM file has to be sent or accessed from CDE for performing the design automation tasks. The possibility to

perform model-based updates that are saved on CDE is something that needs to be extensively researched upon in the future.

If the updation of semantic information and geometry, deletion and creation of elements without creating further clashes and interoperability of the BIM model are possible in a cloud-based design automation system, it would benefit in resolving clashes in the BIM models. This would save time and cost by quickly resolving clashes without the need for a desktop BIM authoring application. An example case would be that: the size change of an HVAC element (pipe) caused a physical clash with a steel beam. Then the architect or structural engineer, on identifying the clash, sent a change request to lower the position of the HVAC element. It could be easily implemented given a fully developed cloud-based design automation system. An approach would be that: HVAC engineers open the model from the CDE and lower the height in the web-based platform. Also, automatic suggestions based on ML or AI-based algorithms executed in cloud-based design automation for clash resolution would benefit the AEC industry to the next level.

The lack of proper guidelines while implementing the design automation tasks is a challenge that needs further investigation. Without a standard workflow and a set of rules, it might be difficult for the users to adapt the web application for their needs. Therefore, the aspects of standardization in designing such automation tasks in the industry or in a company, which is fundamentally formulating a set of rules, requirements, constraints on addressing a specific engineering problem and finding optimal rational boundaries, should be researched further. Further improvements in ICT technologies are expected to solve most of the existing cloud-based design automation limitations by improving the speed and security, solving latency issues, reducing the costs etc. and also could pave the way for new opportunities in design automation capabilities.

References

Afsari, K., Eastman, C., & Shelden, D. (2016). Cloud-Based BIM Data Transmission: Current Status and Challenges. In A. Sattineni, S. Azhar, & D. Castro (Eds.), *Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC), Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC).* International Association for Automation and Robotics in Construction (IAARC). https://doi.org/10.22260/ISARC2016/0129

Allplan Bimplus. (2021, December 3). https://www.allplan.com/products/allplan-bimplus/

Alreshidi, E., Mourshed, M., & Rezgui, Y. (2018). Requirements for cloud-based BIM governance solutions to facilitate team collaboration in construction projects. *Requirements Engineering*, *23*(1), 1–31. https://doi.org/10.1007/s00766-016-0254-6

Authentication (OAuth). (2021). https://forge.autodesk.com/en/docs/oauth/v2/developers_guide/overview/

Autodesk. (2021a, November 29). *What Is Autodesk® BIM 360™?* | *BIM 360* | *Autodesk Knowledge Network*. https://knowledge.autodesk.com/support/bim-360/learn-explore/caas/CloudHelp/cloudhelp/ENU/About-BIM360/files/GUID-A4AF6DE0-3BE4-4CF4-9C84-C780A870D5E2-html.html

Autodesk. (2021b, December 3). *Insight 360*. https://insight360.autodesk.com/oneenergy

Autodesk Forge. (2021a, October 12). https://learnforge.autodesk.io/#/environment/rundebug/2legged

Autodesk Forge. (2019a). *Subscription & Cloud Credit Pricing*. https://forge.autodesk.com/pricing

Autodesk Forge. (2019b). *What is Autodesk Forge?* https://forge.autodesk.com/blog/what-autodesk-forge

Autodesk Forge. (2020). *Design Automation V2 API*. https://forge.autodesk.com/blog/design-automation-v2-api-retirement

Autodesk Forge. (2021b). *APIs | Autodesk Forge*. https://forge.autodesk.com/en/docs/

Autodesk Forge. (2021c). *Model Derivative API | Autodesk Forge*. https://forge.autodesk.com/en/docs/model-derivative/v2/developers_guide/overview/

Autodesk FormIt. (2021, November 29). https://formit.autodesk.com/

Avanova. (2021, December 3). *NOVA Building IT - free thinker with tradition*. https://avanova.de/about

Bello, S. A., Oyedele, L. O., Akinade, O. O., Bilal, M., Davila Delgado, J. M., Akanbi, L. A., Ajayi, A. O., & Owolabi, H. A. (2021). Cloud computing in construction industry: Use cases, benefits and challenges. *Automation in Construction*, *122*, 103441. https://doi.org/10.1016/j.autcon.2020.103441

Bew, M., & Richards, M. (2021, November 10). *Bew-Richards BIM Maturity Model*. https://www.researchgate.net/publication/279293516_BIM_Client_Maturity_Literature_Review/figures?lo=1

Borrmann, A., Köni, M., Koch, C., & Beetz, J. (2015). *Building information modeling: Technologische Grundlagen und industrielle Praxis / Andre Borrmann, Markus Köni, Christian Koch, Jakob Beetz, Herauseber*. *VDI-Buch*. Springer Vieweg.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2018). *Building Information Modeling*. Springer International Publishing. https://publications.cms.bgu.tum.de/books/bim_2018/01_Introduction_06.pdf https://doi.org/10.1007/978-3-319-92862-3

Cascading Style Sheets. (2021). https://en.wikipedia.org/w/index.php?title=CSS&oldid=1054554214

Chen, H.-M., Chang, K.-C., & Lin, T.-H. (2016). A cloud-based system framework for performing online viewing, storage, and analysis on big data of massive BIMs. *Automation in Construction*, *71*, 34–48. https://doi.org/10.1016/j.autcon.2016.03.002

Dassault Systèmes. (2021). *END-TO-END COLLABORATION ENABLED BY BIM LEVEL 3: An Industry Approach Based on Best Practices from Manufacturing.* Dassault Systèmes.

Data Management API. (2021, November 13). https://forge.autodesk.com/en/docs/data/v2/developers_guide/overview/

Design Automation API. (2021, November 14). https://forge.autodesk.com/en/docs/design-automation/v3/developers_guide/overview/

Dixon, J. R. (1995). Knowledge-Based Systems for Design. *Journal of Mechanical Design*, *117*(B), 11–16. https://doi.org/10.1115/1.2836444

Document Object Model. (2021, November 16). https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

Du Juan, & Zheng, Q. (2014). Cloud and Open BIM-Based Building Information Interoperability Research. *Journal of Service Science and Management*, *07*(02), 47–56. https://doi.org/10.4236/jssm.2014.72005

Eastman, C. (1974). An Outline of the Building Description System. https://files.eric.ed.gov/fulltext/ED113833.pdf

Eastman, C., Teicholz, P., Sacks, R., & Liston, K. (2008). *BIM Handbook*. John Wiley & Sons, Inc. https://doi.org/10.1002/9780470261309

Elban, C. (2020, August 28). The History, Evolution and Future of SaaS(Software as a Service). *SaaS Metrics*. https://saasmetrics.co/the-history-evolution-and-future-of-saassoftware-as-a-service/

Eurostat. (2021, November 28). *Cloud computing for business yet to go mainstream in the EU*. https://ec.europa.eu/eurostat/web/products-eurostat-news/-/ddn-20210121-1

Graphisoft. (2021, December 3). *BIMcloud*. https://graphisoft.com/de/teamwork/bimcloud?gclid=Cj0KCQiAnaeNBhCUARIsABEee8Wk34eEx4EYv-TbkDNsUiUaNLQe2gdiXvrs3t2ebSl-4m21zlKqvuEaAlQ8EALw_wcB

Janne Kuuskeri. (2014). *Engineering Web Applications: Architectural Principles for Web Software.* https://doi.org/10.13140/RG.2.1.2457.4889

Jiao, Y., Zhang, S., Li, Y., Wang, Y., & Yang, B. (2013). Towards cloud Augmented Reality for construction application by BIM and SNS integration. *Automation in Construction*, *33*, 37–47. https://doi.org/10.1016/j.autcon.2012.09.018

Kaluža, M., & Vukelić, B. (2018). Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta U Rijeci*, *6*(1), 261–282. https://doi.org/10.31784/zvr.6.1.19

Ma, L., & Sacks, R. (2016). A Cloud-Based BIM Platform for Information Collaboration. In A. Sattineni, S. Azhar, & D. Castro (Eds.), *Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC), Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC).* International Association for Automation and Robotics in Construction (IAARC). https://doi.org/10.22260/ISARC2016/0070

Macdonald, J. (2011). BIM- Adding value by assisting collaboration.

MacMillen, D., Camposano, R., Hill, D., & Williams, T. W. (2000). An industrial view of electronic design automation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *19*(12), 1428–1448. https://doi.org/10.1109/43.898825

Mahamadu, A.-M., Mahdjoubi, L., & Booth, C. Challenges to BIM-Cloud Integration: Implication of Security Issues on Secure Collaboration. In *2013 IEEE 5th International Conference 12/5/2013* (pp. 209–214). https://doi.org/10.1109/CloudCom.2013.127

Mckinsey. (2019). *McKinsey Global Institute Digitisation Index; 2015 or latest available data*. https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/strategy-in-the-face-of-disruption-a-way-forward-for-the-north-american-building-products-industry

McKinsey & Company (2019, December 1). The impact and opportunities of automation in construction. *McKinsey & Company*. https://www.mckinsey.com/business-functions/operations/our-insights/the-impact-and-opportunities-of-automation-in-construction

Mell, P. M., & Grance, T. (2011). The NIST definition of cloud computing. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Mesbah, A., & van Deursen, A. (Eds.) (2007). *Migrating Multi-page Web Applications to Single-page AJAX Interfaces*.

Microsoft Azure. (2021). *What Is Cloud Computing? A Beginner's Guide | Microsoft Azure*. https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

Model Derivative API. (2021). https://forge.autodesk.com/en/docs/model-derivative/v2/developers_guide/overview/

Oesterreich, T. D., & Teuteberg, F. (2016). Understanding the implications of digitisation and automation in the context of Industry 4.0: A triangulation approach and elements of a research agenda for the construction industry. *Computers in Industry*, *83*, 121–139. https://doi.org/10.1016/j.compind.2016.09.006

Papadopoulos, G. A., Wojtkowski, W., Wojtkowski, G., Wrycza, S., & Zupancic, J. (Eds.). (2010). *SpringerLink Bücher*. *Information Systems Development: Towards a Service Provision Society*. Springer US. http://swbplus.bsz-bw.de/bsz31422257xcov.htm https://doi.org/10.1007/b137171

Porwal, A., & Hewage, K. N. (2013). Building Information Modeling (BIM) partnering framework for public construction projects. *Automation in Construction*, *31*, 204–214. https://doi.org/10.1016/j.autcon.2012.12.004

ProjectWise Web. (2021, December 3). https://www.bentley.com/en/products/product-line/project-delivery-software/projectwise-web

Redmond, A., Hore, A., Alshawi, M., & West, R. (2012). Exploring how information exchanges can be enhanced through Cloud BIM. *Automation in Construction*, *24*, 175–183. https://doi.org/10.1016/j.autcon.2012.02.003

RIB Software. (2021, December 3). *iTWO in the Cloud (SaaS) - RIB Software*. https://www.rib-software.com/en/solutions/enterprise-construction-cloud/itwo-in-the-cloud-saas

Sacks, R., Eastman, C., Lee, G., & Teicholz, P. (2018). *Bim Handbook: A guide to building information modeling for owners, managers, designers, engineers, contractors, and facility managers* (Third edition). John Wiley & Sons, Inc. https://onlinelibrary.wiley.com/doi/book/10.1002/9781119287568 https://doi.org/10.1002/9781119287568

Sandberg, M., Gerth, R., Hill, D., Lu, W., Jansson, G., Mukkavaara, J., & Olofsson, T. (2016). Design automation in construction - an overview. *CIB W78 Conference*. http://ltu.diva-portal.org/smash/get/diva2:1045280/FULLTEXT01.pdf

Serrano Mena, A. (2019). *Practical Haskell: A Real World Guide to Programming* (2nd ed.). Apress L. P. https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=5771292

Stokes, M. (Ed.). (2001). *Managing engineering knowledge: Moka: Methodology for knowledge based engineering applications.*

van Steen, M., & Tanenbaum, A. S. (2016). A brief introduction to distributed systems. *Computing*, *98*(10), 967–1009. https://doi.org/10.1007/s00607-016-0508-7

Vouk, M. A. (2008). Cloud Computing Issues, Research and Implementations. *Journal of Computing and Information Technology*, *16*(4), 235. https://doi.org/10.2498/cit.1001391

Wang, X., Wong, J., Li, H., & Chan, G. (2014). A review of cloud-based BIM technology in the construction sector, *19*, Article (ISSN: 1874-4753), 281–291. https://www.itcon.org/papers/2014_16.content.06672.pdf

Web App Development. (2021). https://trio.dev/blog/web-app-development

WHATWG. (2021, November 30). *HTML Standard*. https://html.spec.whatwg.org/multipage/introduction.html#background

WordReference. (2021, December 28). *automation - WordReference.com Dictionary of English*. https://www.wordreference.com/definition/automation

Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, *1*(1), 7–18. https://doi.org/10.1007/s13174-010-0007-6

Zheng, J. (2018). Analysis of collaborative design and construction collaborative mechanism of cloud bim platform construction project based on green computing technology. *Journal of Intelligent & Fuzzy Systems*, *34*(2), 819–829. https://doi.org/10.3233/JIFS-169375

Annex A

Design automation capable Revit plugin development

The web application developed as part of cloud-based design automation runs on the web browser, similar to Revit plugin that runs on on-premise Autodesk Revit. A design automation capable Revit plugin has to be developed as part of web application development, and actually, this design automation capable Revit plugin is what gets executed on the Revit engine on the cloud as part of the Autodesk Forge ecosystem.

When the design automation task (*WorkItem*) has been sent to the design automation API, this will go into a queue of tasks in the server. While the task is executed when it reaches the top of the queue, all the input data must be available at the start. The input data includes the input file such as the .rvt file on which the design automation task is performed and input arguments for the *Activity* such as new parameter values of the selected element. For each design automation task, a design automation capable Revit plugin has to be developed similar to the Revit plugin developed for traditional on-premise design automation. The add-in or plugin can be developed for performing single tasks or multiple tasks according to the architecture of the plugin, but each input BIM model is opened only once during a design automation task.

Conventional Revit plugins are external commands (IExternalCommand) or external applications (IExternalApplication). The plugin developed for the cloud-based design automation is a DB application (IExternalDBApplication) which runs without the UI access to Revit Interface. The plugin will have access to all API functions in RevitAPI.dll. However, the plugin will not have access to RevitAPIUI.dll. If the user has an existing Revit plugin, the references to the RevitAPIUI.dll needs to be removed to make it design automation capable plugin

1. **Development of the Revit plugin that performs the design automation tasks locally**

After developing the plugin implementation logic, a sample Revit plugin is developed using .NET Framework 4.8 in Microsoft Visual Studio. The Revit plugin is developed similar to the design automation plugin for testing locally before uploading as a DA plugin. While developing the Revit plugin, similar situations as in DA were recreated. Only DB level data is accessed within the Revit plugin, and inputs for the automation

tasks in the Revit plugin is similar to DA tasks such that all the inputs were provided at the start. Then the Revit plugin is loaded on the desktop-based Autodesk Revit and tested its functionalities to get an overall view. The testing of the functionality using the Revit plugin to delete all the elements of the same type of the element with the provided ElementID as input in the developed Revit plugin is shown in Figure 44. The limitations with the Revit APIs used for performing automation tasks can be discovered while testing the Revit plugin, and necessary improvements or changes can be made to accommodate the limitations in the plugin logic. This testing acts as a strong base for developing the design automation capable plugin. After the necessary testing and validation, the final version of the plugin could be used for converting to the design automation capable plugin.

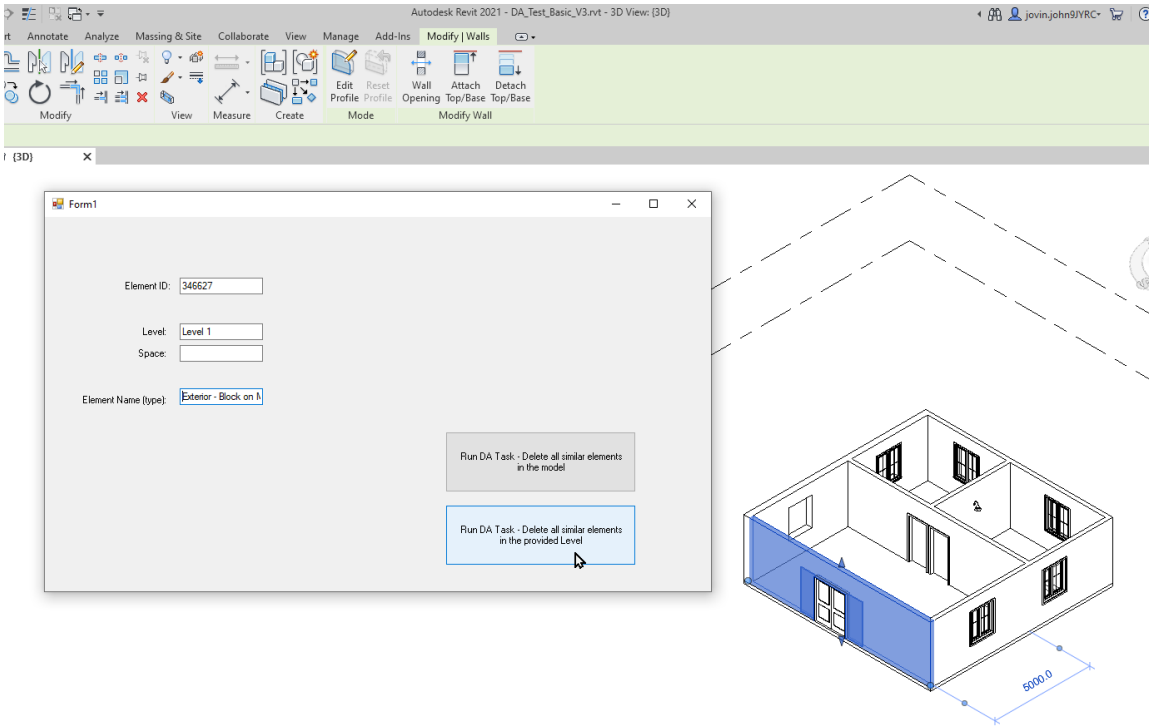

Figure 44: Testing the developed Revit plugin locally

2. Converting from local Revit plugin to design automation capable plugin

In order to convert from the fully functional local Revit plugin, which was well tested and updated, the following steps as documented in (Autodesk Forge, 2021d) can be followed:

- Add a package reference to the DesignAutomationBridge[18] .
- Remove the references to all Revit UI elements.
- Convert IExternalApplication or IExternalCommand to IExternalDBApplication
- Add an event handler for DesignAutomationReady, which raises the event DesignAutomationReadyEvent defined in DesignAutomationBridge when it is ready to run the add-in. This handler is the entry point to the code.
- Handle failures encountered while performing design automation tasks using default error handler provided by DesignAutomationBridge to communicate warnings and errors in the absence of UI.
- Build the add-in to prepare design automation capable Revit add-in

User interface (UI) development

Preliminary UI based on the Autodesk code sample

Figure 45: Example UI provided by Autodesk Forge: the development of prototype UI started inspired by this basic layout (Autodesk Forge, 2021a)

[18] DesignAutomationBridge.dll is a Autodesk library that contains all the functionalites to interface with the Design autoamtion
URL: https://www.nuget.org/packages/Autodesk.Forge.DesignAutomation.Revit

Initial UI

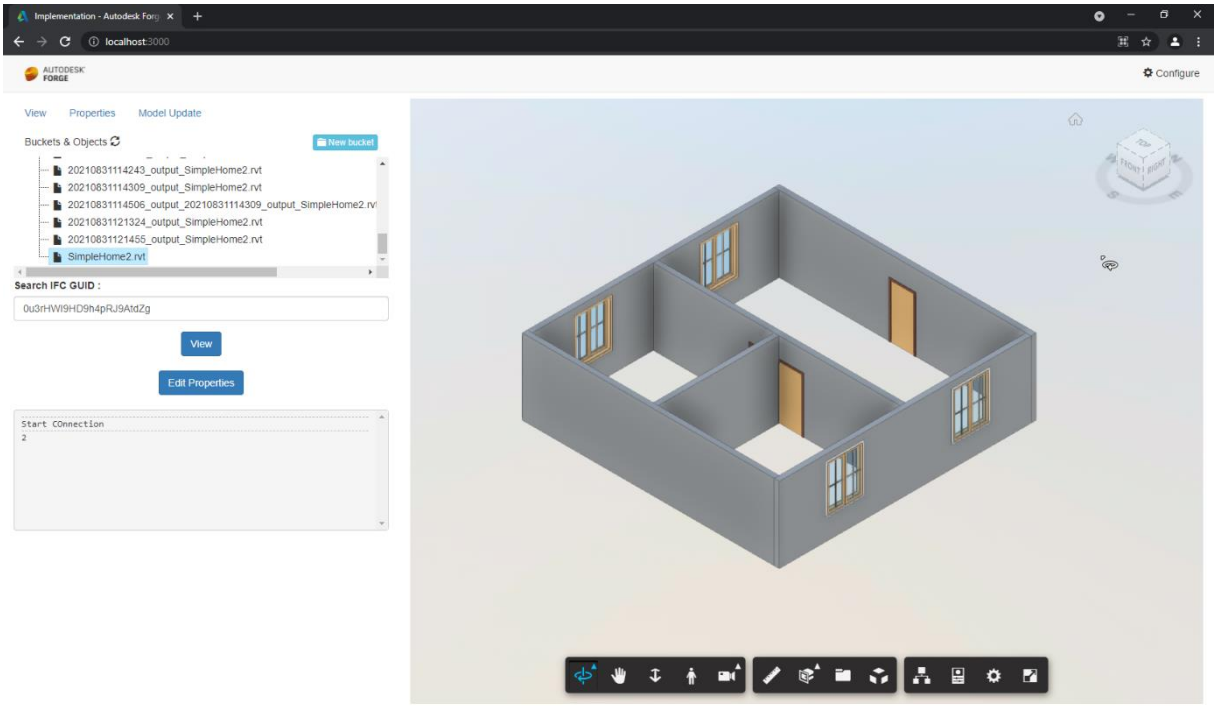

Figure 46: Initial UI of the developed prototype

Final UI

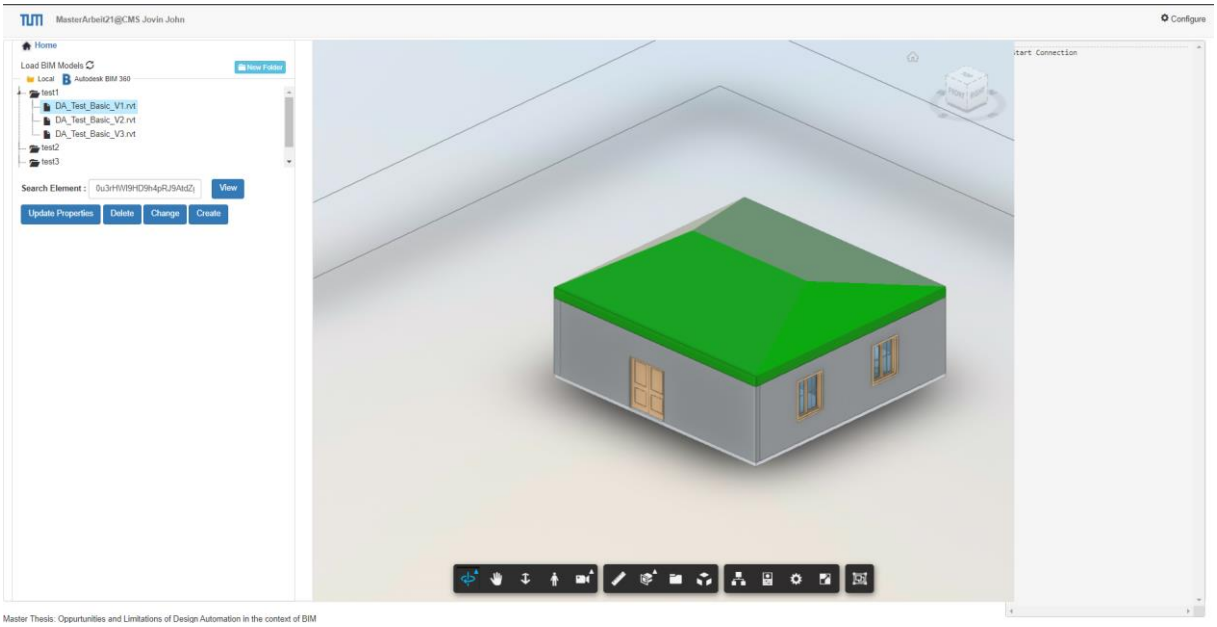

Figure 47: Final UI of the developed prototype