Project Report

Real Time Collaboration on Building Models

Submitted by

Anjana Muraleedharan Nair — Matriculation Number:125512

Prasamsa Nandanan — Matriculation Number: 125288

Selva Ganapathy Elangovan — Matriculation Number: 124769

Vishal Sanjay Shivam — Matriculation Number: 125353

Submitted to

Chair of Intelligent Technical Design

First Referee: Prof. Dr. Christian Koch

Second Referee: Junior-Prof. Mr. Mathias Artus

DECLARATION

We affirm that our team is solely responsible for creating this report, and it has not been previously submitted, either in its entirety or partially, for any coursework or degree requirements. Unless explicitly mentioned through references or acknowledgments, the content presented is entirely our original work.

Weimar, 31.03.2024. Signature

Anjana Muraleedharan Nair

Prasamsa Nandanan

Selva Ganapathy Elangovan

Vishal Sanjay Shivam

ABSTRACT

Real-Time Collaboration on Building Models addresses the critical need for seamless communication and collaboration within architectural design and building information modeling. This project introduces a framework utilizing serialized XML objects to facilitate real-time updates to Resource Description Framework data, a pivotal technology in the structured representation of information for the construction industry. The source system initiates modifications to RDF data, encompassing actions such as updating entity names, integrating new entity, and defining property sets. These modifications are serialized into XML objects and transmitted to a target system through a network connection. The target system actively listens for incoming updates, extracting metadata and queries to implement changes within its RDF database. Furthermore, the system extends its collaborative reach by propagating updates to other target systems. The project underscores the effectiveness of communication and synchronization mechanisms in RDF data, fostering a collaborative ecosystem for professionals in architecture, engineering, and stakeholders engaged in the intricacies of building modeling. Undertaken to address industry challenges, the project aims to deliver a robust solution, enhancing efficiency in architectural design and construction processes.

ACKNOWLEDGEMENT

We chose to complete this section of our project report at the eleventh hour. Having dedicated several months to the project, overcoming various challenges along the way, we anticipated gaining valuable experiences.

We extend our sincere thanks to Junior Professor Mathias Artus for generously contributing his time, energy, and exemplary guidance throughout the project. His expertise and profound understanding of the subject greatly enriched our work. Special appreciation goes to the faculty of Intellectual Design at Bauhaus Universität for their continuous support and advice.

Our gratitude extends to our peers and colleagues for providing a conducive environment for engaging in discussions and brainstorming sessions. We also express profound thanks to our family and friends, whose unwavering love and support served as a constant source of motivation, enabling us to approach this project with a clear mind.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

IFC	Industry Foundation Classes
RDF	Resource Description Framework
TCP	Transmission Control Protocol
BIM	Building Information Modelling
XML	Extensible Markup Language
SPARQL	SPARQL Protocol and RDF Query Language
AEC	Architecture, Engineering, and Construction
ACC	Automated Compliance Checking
NLP	Natural Language Processing
BIMQL	Building Information Model Query Language
CDE	Common Data Environment
UML	Unified Modeling Language
BPMN	Business Process Model and Notation

1. INTRODUCTION

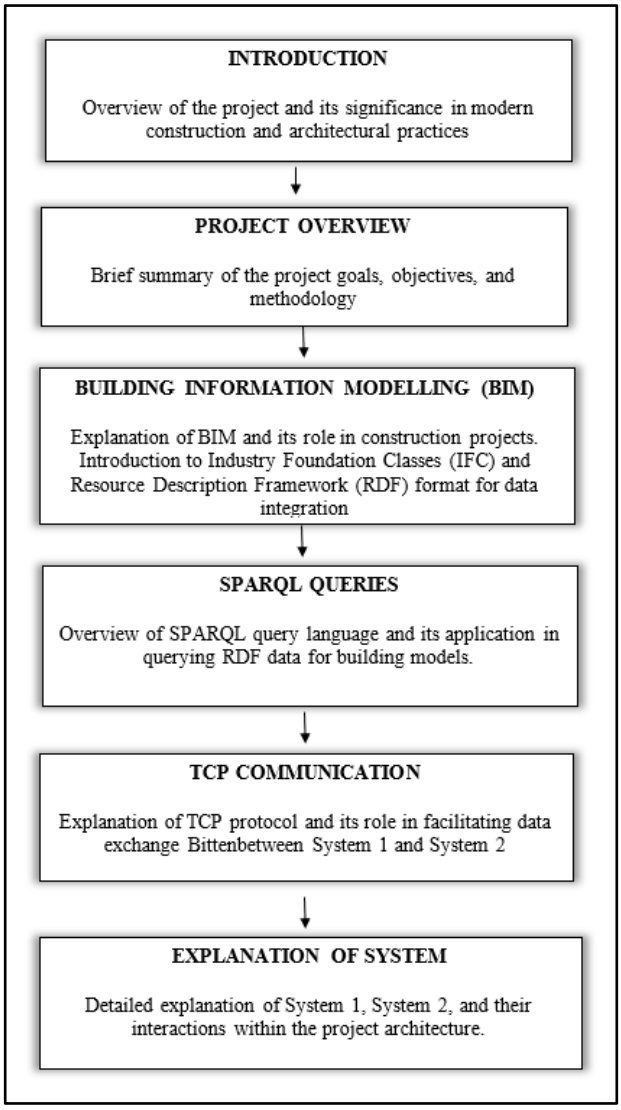

Figure 1 Structure of the Project Introduction

In today's data-driven landscape, the need for accurate and timely information dissemination across diverse systems is paramount. The requirements of contemporary applications are frequently not met by traditional manual approaches to data administration, which results in errors, inconsistencies, and inefficiencies. Acknowledging this difficulty, the "Real Time Collaboration on Building Models" project seeks to automate, streamline, and improve the system's dependability to transform data management procedures. The project seeks to enhance the process of updating and distributing data across multiple systems with ease by utilizing cutting-edge technologies like RDF for data representation, IFC for interoperability, SPARQL queries for data retrieval, XML serialization and

deserialization for data interchange, and TCP listeners for network communication. Fundamentally, this project aims to provide a solid and scalable solution that guarantees data integrity, consistency, and accessibility across several platforms and applications, thereby addressing the inherent complexity and constraints of manual data management.

In the context of evolving technology landscapes and increasingly complex data ecosystems, the need for a sophisticated data management solution becomes apparent. Traditional methods of data management, reliant on manual intervention and human oversight, are inherently prone to errors and inefficiencies. As organizations grapple with vast volumes of data generated at an unprecedented rate, the inadequacy of manual processes becomes more pronounced. Against this backdrop, the project "Real Time Collaboration on Building Models" emerges as a proactive response to the challenges posed by manual data management. By automating critical aspects of data update and distribution, this project aims to enhance operational efficiency, reduce errors, and ensure the seamless flow of information across disparate systems.

1.1 Project Overview

The primary objective of the "Real Time Collaboration on Building Models" project is to develop a comprehensive solution that automates the process of updating and distributing data across multiple systems while leveraging advanced technologies such as RDF, IFC, SPARQL queries, XML serialization and deserialization, and TCP listeners. By harnessing these technologies, the project aims to streamline data management processes, enhance operational efficiency, improve data accuracy and consistency, and reduce the risk of errors and inconsistencies. Additionally, the project seeks to increase organizational agility and responsiveness by enabling timely updates and ensuring the availability of accurate information when needed.

The scope of the initiative includes retrieving, transforming, updating, and distributing data, among other components of the data management lifecycle. Leveraging RDF allows for the standardized representation and connection of data across diverse systems, ensuring seamless integration and interoperability. IFC facilitates the exchange of building information models between different software applications, promoting interoperability within the Architecture, Engineering, and Construction industry. SPARQL queries enable targeted data retrieval from RDF-formatted data, enhancing exploration and analysis capabilities within the BIM context. XML serialization and

deserialization facilitate the interchange of structured data between systems, while TCP listeners enable efficient network communication for data transmission.

The project adopts an iterative and agile development approach, characterized by continuous feedback, collaboration, and adaptation. Requirements gathering and system architecture design are informed by the need to leverage RDF, IFC, SPARQL queries, XML serialization and deserialization, and TCP listeners to achieve project objectives. The development phase involves the implementation of software components that utilize these technologies for data management, integration with existing systems, and testing to ensure functionality and reliability. Throughout the project lifecycle, emphasis is placed on transparency, communication, and stakeholder engagement to ensure alignment with organizational objectives and user needs.

The "Real Time Collaboration on Building Models" project holds significant implications for organizations seeking to optimize their data management processes. By leveraging RDF, organizations can benefit from standardized data representation and integration, facilitating seamless interoperability across diverse systems and applications. IFC enables efficient exchange of building information models, promoting collaboration and interoperability within the AEC industry. SPARQL queries enhance data exploration and analysis capabilities, enabling informed decision-making within the BIM context. XML serialization and deserialization facilitate the interchange of structured data, while TCP listeners ensure efficient network communication for data transmission. Overall, the project's significance lies in its ability to empower organizations with the tools and technologies needed to thrive in an increasingly data-driven world while addressing the challenges of manual data management.

1.2 Building Information Modeling

In contemporary construction and architectural endeavors, Building Information Modeling stands as a cornerstone, facilitating superior planning, collaboration, and management of structures. BIM involves digitally representing a structure's physical attributes, integrating architecture, engineering, and construction elements. Notably, Autodesk's Revit emerges as a leading platform for BIM implementation, offering stakeholders 3D modeling, collaboration, and documentation capabilities. To ensure seamless interoperability across diverse software applications within the AEC industry, Industry Foundation Classes, an open file format, play a pivotal role. IFC enables the exchange of

building information models, encompassing crucial data on building elements, geometry, properties, and attributes, thus facilitating smooth information interchange throughout the project lifecycle. Furthermore, to enhance interoperability and integration, IFC files are frequently converted into Resource Description Framework format. RDF provides standardized data representation and connectivity across the web, allowing for the seamless integration of building information with various datasets and applications. Its capabilities enable the effortless merging of data schemas, support schema evolution, and enable straightforward naming of relationships between entities.

1.3 SPARQL Queries

SPARQL emerges as a pivotal tool for querying and manipulating data stored in RDF format. RDF, with its data model representing information in subject-predicate-object triples, aptly suits the representation of diverse and interconnected data, especially within domains like the Semantic Web and building information modeling. Within System 1, SPARQL queries play a crucial role in updating an RDF triple store with fresh information about building elements. For instance, the `UpdateRdfDatabase` function likely executes SPARQL update commands extracted from serialized XML received by the data processing system. These commands might entail inserting, deleting, or modifying RDF triples in the triple store based on the instructions provided. Additionally, SPARQL facilitates querying RDF data in the query generation system to extract relevant information about building elements, properties, and relationships. This extracted data can then be further processed, serialized, and transmitted to other systems, thereby contributing to broader data integration and interoperability efforts. SPARQL enables the querying of building data structured in RDF format, facilitating the retrieval of targeted information from diverse data sources. It supports various query patterns such as required and optional graph patterns, providing flexibility in data retrieval. Furthermore, SPARQL query results can manifest as either result sets or RDF graphs, ensuring seamless integration and compatibility within BIM contexts, thereby enhancing the efficiency and interoperability of building information modeling systems.

1.4 TCP (Transmission Control Protocol)

Within the realm of network communication protocols, the Transmission Control Protocol stands as a cornerstone, operating at the transport layer of the TCP/IP suite. Its primary function is to facilitate reliable and connection-oriented communication between devices over IP-based networks. TCP

ensures the integrity of data transmission through various mechanisms such as acknowledgments, retransmissions, and sequencing, thereby guaranteeing that data sent from one endpoint is received correctly and orderly by the destination endpoint. In the context of the project, TCP plays a pivotal role as the communication protocol enabling seamless data exchange between two critical systems: the Query Generation System (System1) and the Data Processing System (System 2). This bidirectional communication channel is established through TCP clients and servers integrated into the codebase. Serialized XML data containing updates to the Resource Description Framework database is reliably transmitted from the Query Generation System to the RDF Processing System via TCP. The TCP client takes the initiative in establishing connections and transmitting data packets encapsulating serialized XML information. On the other end, the TCP server, upon accepting connections, reads and processes the received data, thus facilitating the updating of the RDF database. This synchronized communication ensures the efficient synchronization of crucial information essential for the project's functionality.

Moreover, the bidirectional nature of TCP communication ensures that once the RDF database is updated in the Data Processing System, acknowledgments regarding the updates are promptly sent back to multiple client systems in real-time, further enhancing the reliability of communication across the network. In this report, the intricate workings of TCP as a fundamental communication protocol and its indispensable role in facilitating efficient data exchange between the Query Generation System and the Data Processing System are explored, ultimately contributing to the seamless operation and synchronization of vital information within the project framework.

1.5 Explanation of Systems

System 1 operates as the Query Generation System entrusted with managing and updating RDF data pertinent to building models. It harnesses technologies such as RDF, XML serialization, and TCP listeners to adeptly handle data updates and distribution. Conversely, System 2, referred to as the Data Processing System, processes updates received from System 1 utilizing SPARQL queries, subsequently updating the RDF database as necessary. Additionally, acknowledgments regarding the updates are communicated back to System1 via TCP communication channels, thereby ensuring the seamless flow of information. While not explicitly detailed, Target Systems (System 3) likely represents another integral component or system within the broader project architecture, contributing to its overall functionality and efficiency.

By incorporating these technologies and methodologies into the project, "Real-Time Collaboration on Building Models" aims to revolutionize data management in the AEC (Architecture, Engineering, and Construction) industry. This initiative enables seamless collaboration, improved decision-making, and enhanced efficiency throughout the project lifecycle. With the implementation of TCP for reliable data exchange and bidirectional communication between the Query Generation System (System 1) and the Data Processing System (System2), the project ensures that all relevant parties are promptly informed about successful updates. This enhances the overall reliability and effectiveness of the communication process, thereby facilitating smoother workflows and fostering innovation in the AEC domain.

2. LITERATURE REVIEW

Throughout the project duration, an extensive review of papers, articles, and textbooks has been conducted, providing valuable insights into operational principles and real-time collaboration aspects in model construction. The sources examined offer a comprehensive understanding of essential concepts in this domain.

2.1Building Information Modeling (BIM):

Semantic Automation for Compliance Checking proposes a semantic approach integrating BIM and NLP for Automated Compliance Checking (ACC) in construction. The findings emphasize the effective streamlining of ACC procedures through NLP-driven rule extraction and semantic mapping. However, limitations are identified, including dependency on data quality, the need for expert intervention in conflict resolution, and challenges in semantic mapping complexity. The study investigates the effectiveness of a semantic approach in automating ACC procedures within construction projects. By leveraging Natural Language Processing to extract rule terms and logic links from regulatory papers, the method simplifies the ACC process. This involves mapping extracted terms to concepts in Building Information Modeling data, providing flexibility and efficiency through automated SPARQL query development. This integration of intelligent technologies, including NLP and semantic enrichment of BIM data, addresses the drawbacks of manual ACC procedures, offering a comprehensive solution for automation.[1]

However, certain limitations must be considered. The method's efficacy heavily depends on the completeness and quality of both BIM data and regulatory documents. Errors in rule interpretation and mapping arising from inaccuracies or inconsistencies in these datasets can impact the reliability of ACC results. Additionally, the complexity of construction laws and the intricate nature of natural language pose challenges in semantic mapping, requiring expert intervention for conflict resolution during information extraction. While the integration of intelligent technologies significantly streamlines the ACC process, achieving complete autonomy remains compromised by the need for human expertise in certain instances. [1]

Another approach presenting a framework for generic semantic enrichment, this paper enhances object geometries in BIM models. The framework showcases effectiveness in practical applications

but encounters limitations in algorithm scalability, adaptability, and scenario-specific adjustments. The research presents a robust foundation for semantic enrichment, encompassing four tasks and associated tools designed to extract detailed semantic information about object geometries. This system is demonstrated in practical application by enhancing Building Information Modeling models, showcasing the utilization of richer semantics from platforms like SketchUp and ArchiCAD to reconstruct apartment models within Revit. The study not only highlights the tangible benefits of semantic enrichment for enhancing BIM data but also emphasizes its role in overcoming interoperability barriers, facilitating seamless model transfers between different BIM software platforms.[2], [3]

However, certain limitations need consideration. Firstly, the performance and adaptability of the developed tools are influenced by their reliance on rule-based algorithms with threshold values that may require manual adjustments. This aspect could impact the tools' effectiveness in diverse scenarios. Secondly, while the tools are primarily tailored for residential scenarios, additional efforts are needed to accommodate a broader range of use cases. Moreover, the method's success at Level of Development (LOD) 200 may pose challenges when scaling up to higher LOD models, introducing complexities and necessitating more comprehensive representations. These limitations highlight areas for refinement and expansion to ensure broader applicability and scalability in real-world BIM implementations. [2], [3]

2.2 SPARQL Query Language:

BIMQL – An Open Query Language for Building Information Models. The BIMQL paper introduces a user-friendly query language for IFC-based models, enhancing query capabilities in BIM workflows. While highlighting improved integration and scalability, challenges include the need for further implementation and testing, expanding natural language capabilities, and enhancing integration with Model View Definition templates.

The development of BIMQL aimed to address the demand for a user-friendly query language tailored to diverse information needs within the building design and construction process. It facilitates the flexible and ad hoc generation of partial data sets from IFC-based models, streamlining complex searches and enhancing efficiency for users interacting with building information models. BIMQL's design emphasizes improved query capabilities, allowing users to choose and edit various

aspects of models using more natural language construction and domain-specific shortcuts, making it adaptable to the unique requirements of different stakeholders. [4]

Despite its promising features, certain limitations need consideration. While BIMQL has showcased its integration into current BIM software environments on the bimserver.org platform and demonstrated scalability in handling queries on models of different sizes, full implementation and extensive testing across diverse software systems and modeling scenarios remain unfinished. Additionally, BIMQL currently exhibits limited natural language capabilities, suggesting the need for further development to enhance accessibility for individuals without extensive technical knowledge of IFC model schema structures. Furthermore, the integration with Model View Definition (MVD) templates for more complex query patterns and information aggregation is not fully realized in the current version, leaving room for future iterations to refine and expand its capabilities.[4]

An Approach of Automatic SPARQL Generation for BIM Data Extraction. Addressing SPARQL query generation, this paper automates the process for BIM data extraction. The approach proves effective in case studies but faces challenges in query breadth due to keyword matching and handling complex relationships, depending on semantic web technologies.The proposed approach offers a significant advancement in the automation of SPARQL query generation for extracting BIM data, alleviating the challenges associated with manual query crafting. By leveraging basic search keywords and their relationships within the BIM instance model, the method efficiently produces SPARQL queries, enhancing the accessibility and overall efficiency of working with BIM data. Case studies demonstrate the accuracy and effectiveness of the resulting queries in extracting relevant information from BIM models, affirming the approach's suitability for a range of engineering and BIM data analysis applications, eliminating the need for users to possess in-depth knowledge of SPARQL or the underlying BIM schema. [5]

However, certain limitations warrant consideration. The method's reliance on matching phrases in the BIM knowledge base with user-entered query keywords, predominantly through string matching, may constrain the breadth and effectiveness of inquiries if the keywords do not closely align with the phrases in the BIM database. Additionally, for queries involving complex relationships among multiple targets, the procedure may generate numerous path query files, posing challenges in

manually identifying the shortest path connecting all keywords. This complexity implies that additional programming efforts may be necessary to adequately handle and interpret these files. Furthermore, the approach's dependency on semantic web technologies, specifically the Stardog RDF database for path searches, may limit its applicability on certain platforms, necessitating consideration of alternative solutions for broader implementation.[5]

2.3 Parallel Programming:

Parallel Genetic Algorithms for Optimizing Resource Utilization. Introducing a parallel genetic algorithm framework, this paper improves resource utilization in large-scale construction projects. While showcasing efficiency and effectiveness, challenges include the need for further research to confirm generalizability, optimization of parameters, and potential difficulties in managing parallel computing resources. In large-scale construction projects, the parallel genetic algorithm framework improves resource usage efficiency. It shows usefulness on limited processors, but more research is needed to determine generalizability and ideal parameter values. [6]

The designed parallel genetic algorithm framework exhibits significant gains in computational efficiency, particularly in maximizing resource usage for large-scale building projects. Through the utilization of a cluster of parallel processors, the framework demonstrates a substantial reduction in the time required for optimization tasks, emphasizing its efficiency in real-world construction engineering and management offices, even with a limited number of parallel processors. The study provides a comprehensive assessment of the framework's practicality, showcasing its potential to address challenges related to computing power shortages in construction project scenarios. Additionally, a comparative analysis of global and coarse-grained parallel genetic algorithms within the framework reveals insights into their respective performances, highlighting trade-offs between computational time, effectiveness, and solution quality.[6]

However, certain limitations require consideration. The generalizability of the framework's performance, as observed in experiments limited to specific project sizes and configurations within the construction industry, necessitates further investigation to confirm its applicability across a broader range of project sizes and types. The research also underscores the impact of setup parameters, such as migration rate and interval, on the performance of the coarse-grained parallel genetic algorithm, highlighting the need for ongoing investigation to determine optimal criteria for

various project circumstances. Despite the framework's practical design, challenges may arise in construction management offices regarding the management of parallel computing resources and multiple processors, particularly concerning infrastructure and technical expertise. Addressing these limitations will be crucial for ensuring the broader effectiveness and applicability of the parallel genetic algorithm framework in diverse construction management contexts.[6]

2.4 Database Integration:

Cloud-Based BIM Governance Platform Requirements and Specifications in this study outlines requirements for the Govern BIM platform, emphasizing a cloud-based architecture. Challenges involve the lack of empirical support, user adoption considerations, and insufficient discussion on security and privacy issues related to construction project data. The research successfully identifies the critical requirements and specifications essential for constructing the GovernBIM platform, focusing on domain-specific, non-functional, and functional aspects crucial for effective teamwork and BIM data management during construction projects. The integration of BPMN for business process modeling and UML for system and structural modeling provides a comprehensive framework for defining GovernBIM, allowing for detailed descriptions of use cases, governance architecture, and internal and external business processes. Additionally, the study emphasizes the practical application of a well-organized cloud-based architecture, leveraging the scalability, stability, and accessibility of cloud computing to enhance BIM governance systems.[7]

However, certain limitations warrant consideration. The paper primarily outlines the theoretical foundation and specifications of the GovernBIM platform, lacking detailed insights into its real-world implementation and empirical validation. Future investigations should focus on implementing and assessing the platform across various construction projects to evaluate its efficiency and expandability. The study briefly addresses user adoption concerns and the learning curve associated with BPMN and UML, emphasizing the importance of providing intuitive interfaces and reducing complexity for end users. Furthermore, while acknowledging the advantages of cloud technologies, the study does not delve into the security and privacy issues inherent in cloud-based BIM governance. Future research should address the implementation of robust security and privacy standards within GovernBIM, given the sensitive nature of construction project data.[7]

2.5 Serialization and Deserialization Approaches:

In the realm of Building Information Modeling (BIM) advancements, diverse methodologies have been explored to enhance data extraction and query processes. One such innovative approach involves a two-stage Text-to-BIMQL semantic parsing utilizing Graph Neural Networks.[8]This method demonstrates remarkable achievements in precision, scalability, and real-world applicability, especially excelling in relational activities and entity linking within BIM information extraction. However, it acknowledges challenges related to dataset diversity and suggests avenues for improvement in handling more intricate queries.

Parallelly, another avenue of progress in BIM data retrieval is explored through an NLP-Based Query-Answering (QA) System.[9] This system, leveraging Natural Language Understanding (NLU), Information Extraction (IE), and Natural Language Generation (NLG) modules, achieves an impressive accuracy rate of 81.9% in responding to natural language questions related to BIM models. While contributing significantly to enhanced efficacy and accessibility in construction project management, it confronts limitations concerning scalability, generalizability, and the handling of complex queries. These challenges underscore the need for ongoing updates to refine comprehension and address intricate information demands.

Beyond information extraction and query systems, a unique contribution to the BIM landscape is presented in the form of a Novel Cost Model of XML Serialization.[10] Focused on understanding the serialization cost's impact on software systems, particularly within service-oriented architecture (SOA), the model offers insights into the linear relationship between serialization costs and array length, shedding light on various data types. This model's findings, though illuminating, come with considerations such as platform-specificity and the need for further exploration in varied deployment scenarios. The study's limitations underscore the importance of addressing factors like network latency and communication protocol variances for a comprehensive understanding of XML serialization costs in real-world applications.

These diverse methodologies collectively propel the field forward, each offering unique insights and innovations in BIM data management, extraction, and serialization. The challenges identified in these studies pave the way for future research endeavors to refine and expand upon these approaches, ultimately contributing to a more robust and efficient BIM ecosystem [11].

2.6 Common Data Environment:

The proposed federated Common Data Environment for Issue Management in the building sector emphasizes interoperability and data reusability. This framework integrates project-centric and stakeholder-centric layers, aligning with industry standards such as open CDE APIs and ISO 19650 Maturity Stage 3. It enhances data ownership, maintains a single source of information throughout the digital asset life cycle, and enables discipline-agnostic data organization for various AECO purposes. Challenges include managing data modification control within vaults, ensuring data accessibility and recoverability in case of vault owner failure, and addressing concerns regarding data security and trust among stakeholders [12].

Similarly, the framework for version control in asynchronous Building Information Modeling collaboration introduces a novel approach. It emphasizes a diff-and-patch mechanism and graph representations for tracking object-level changes. This framework proposes an optimistic concurrency framework to manage concurrent modifications, aiming to streamline collaboration and enhance coordination among stakeholders in construction projects. Challenges include the need for further adaptation of existing technologies, practical implementation and integration into existing BIM platforms, and complexities in managing concurrent model states in large-scale projects [13].

Furthermore, an approach for establishing a Distributed Common Data Environment (DCDE) utilizing blockchain and the Interplanetary File System (IPFS) is presented for secure collaborative design in the realm of Building Information Modeling (BIM). This innovative system ensures data integrity through blockchain technology and securely stores design files using IPFS, providing a robust platform for efficient collaboration in BIM projects. Key findings include the proposal of a DCDE framework integrating blockchain and IPFS, the development of a blockchain transaction data model and smart contract to support DCDE functionalities, and testing results demonstrating the feasibility of DCDE for secure design collaboration. Challenges such as determining collaboration workflows and information exchange methods within the integrated DCDE, along with the need for further exploration into implementing blockchain and IPFS for secure collaboration, underscore the importance of ongoing research in this field [14].

Top of Form, the collection of reviewed papers collectively advances the field of Building Information Modeling, addressing common themes and challenges. One prevalent topic revolves

around the integration of BIM with Natural Language Processing (NLP) for Automated Compliance Checking. The semantic approach proposed in the first study streamlines ACC procedures through NLP-driven rule extraction and semantic mapping, emphasizing the need for accurate data sources and expert intervention for conflict resolution [1]. Another common thread is the exploration of SPARQL Query Language for BIM, with the BIMQL paper introducing a user-friendly query language for IFC-based models and an approach automating SPARQL generation for BIM data extraction. While showcasing enhanced query capabilities, these papers stress the importance of further implementation, testing, and consideration of alternative solutions for broader integration.[4], [5]

Parallel programming also emerges as a focal point, with a paper introducing a parallel genetic algorithm framework for optimizing resource utilization in large-scale construction projects. Despite demonstrating efficiency, challenges such as generalizability, parameter optimization, and managing parallel computing resources underscore the need for ongoing research across various project sizes and configurations.[6]. The overarching theme is complemented by a study outlining requirements for a Cloud-Based BIM Governance Platform, emphasizing a cloud-based architecture but recognizing challenges in empirical support, user adoption, and security considerations. Additionally, innovative approaches to BIM data extraction and serialization further contribute to the collective progress in the field, with identified challenges paving the way for future refinements and expansions in BIM methodologies [7], [8], [9], [10], [11]. Additionally, frameworks for federated Common Data Environments (CDEs) [12], version control in asynchronous BIM collaboration [13], and Distributed Common Data Environments (DCDEs) utilizing blockchain and IPFS [14] address data ownership, collaboration efficiency, and security concerns in the building sector. Challenges include managing data modification control, adapting existing technologies, and implementing secure collaboration workflows [12], [13], [14].

2.7 Problem Statement

Despite the advancements in Building Information Modelling technology, updating the IFC files in real-time from Revit is challenging. The conventional methods lack the ability to automatically synchronize the Revit model with the corresponding IFC files resulting in inefficiency in data exchange and collaboration among project stakeholders.

2.8 Objectives

- Automated real-time updates: Develop a solution to automatically update the IFC files in Revit in real-time when the modifications are done to the Revit model.
- Enhance data manipulation: Implement functionalities using XBIM library efficiently manipulate BIM entities within the REVIT model.
- Efficient communication infrastructure: Develop a robust communication infrastructure utilizing TCP Listener to facilitate data exchange between the REVIT model and external systems.

3. METHODOLOGY

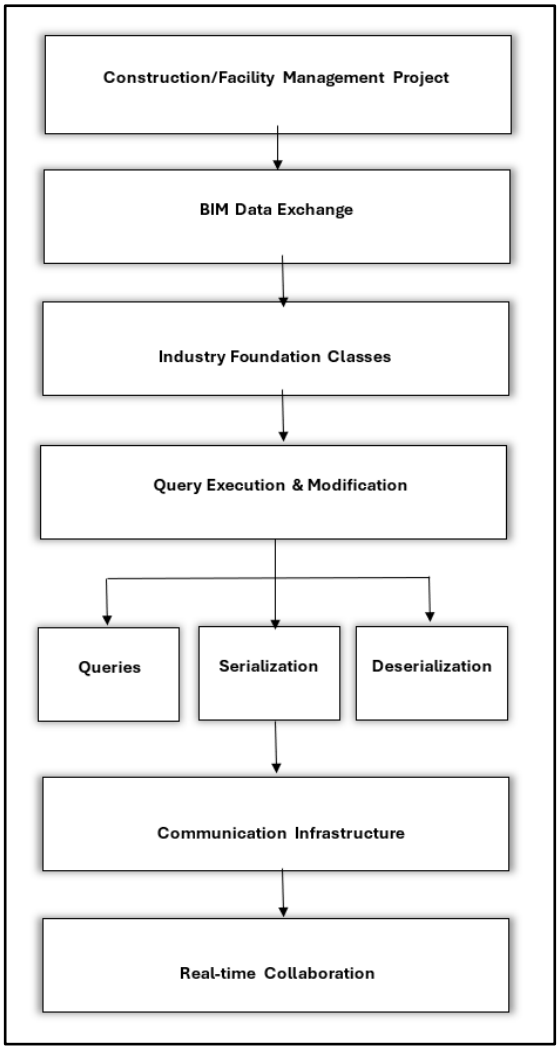

Figure 2 Structure of the Methodology

In the realm of construction and facility management projects, the exchange of Building Information Modeling data among stakeholders is critical for effective collaboration and project success, because it ensures that all stakeholders involved have access to up-to-date and accurate information. The Industry Foundation Classes serve as a standardized framework for exchanging BIM data. They provide a common language and structure that allow different software platforms to communicate seamlessly. This ensures consistency and compatibility across various software platforms. However, ensuring that exchanged data accurately reflects modifications made to the BIM model poses challenges. As projects progress, design changes, additions, and modifications are inevitable. To

address this, a series of approaches has been developed to streamline the process of updating exchanged data automatically.

Queries are employed to modify the exchanged data, with a dedicated class facilitating their execution. Queries allow stakeholders to extract specific information from the BIM model and make targeted modifications to the exchanged data, ensuring accuracy and alignment with the latest project developments.

Furthermore, serialization and deserialization processes are integrated into the workflow to enhance data exchange and synchronization. Serialization converts queries into a suitable format for storage and transmission, while deserialization reverses this process, ensuring compatibility between different systems. These processes facilitate smooth communication and data exchange among stakeholders.

Moreover, a robust communication infrastructure is established to enable real-time collaboration and data exchange among stakeholders. Notifications are sent to clients upon any modifications to the BIM model, triggering the automatic generation and execution of queries to update the exchanged data. This ensures that all stakeholders are promptly informed of changes and that the exchanged data remains up-to-date.

In summary, these approaches offer a reliable solution for automatically updating exchanged BIM data, thereby improving the efficiency and accuracy of information sharing in construction and facility management projects. By leveraging standardized frameworks, query execution, serialization, deserialization, and real-time communication, stakeholders can collaborate seamlessly and ensure the integrity of project data throughout its lifecycle.

3.1 BPMN

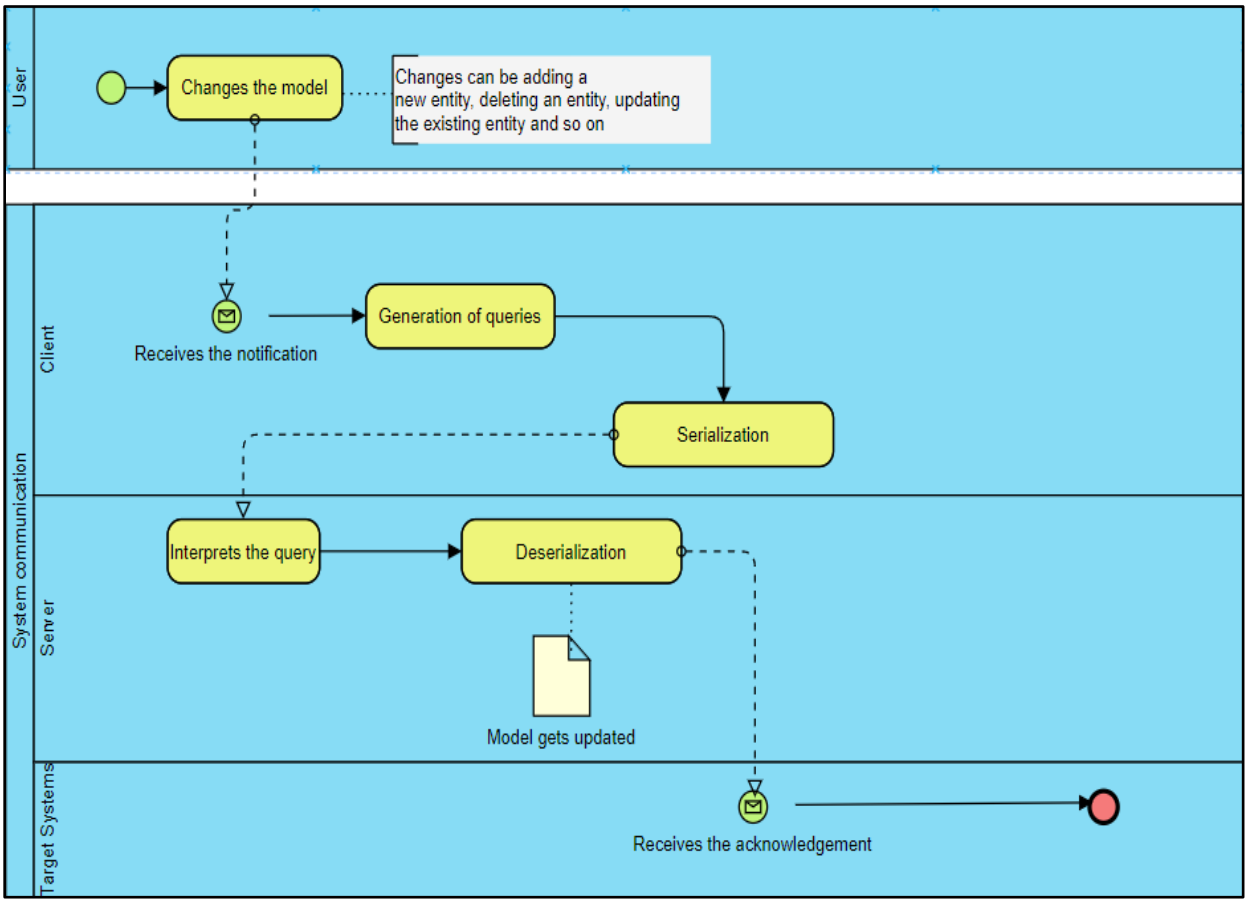

Figure 3 Business Process Modeling Notation (BPMN) Flowchart

The above BPMN diagram employs various elements to represent the process flow, starting with a Start Event indicated by a green circle, signifies the action of the user to change a model. This change is an activity which includes actions like adding, updating, or deleting entities within the system. When the user initiates a change, the notification is communicated to the client through a Message Flow, depicted as dashed lines. This signals that the client, a separate organizational entity within the diagram, has become active in the process.

In the Client side, generation of queries takes place, followed by 'Serialization', which are connected by Sequence Flows - solid lines with arrowheads. These illustrate the order of activities, leading to the serialized queries being sent to the Server via another Message Flow.

On the Server side, deserialization takes place. This is essential for the server to interpret the query. Once interpreted, the model gets updated and reflects the changes initiated by the user. The process concludes with an End Event in the Server, depicted by a red circle with a thick border, representing the server sending an acknowledgment to the target systems.

The BPMN diagram thus facilitates a clear visualization of the communication and operational sequence between the user, client, and server, ensuring that the changes to the model are executed efficiently. It delineates each step with precision, ensuring a shared understanding of the process flow for stakeholders involved in the system's operation.

3.2 Use Cases

1. Modify the name of an existing wall.
2. Set the name of a newly created wall.
3. Add owner history information to an existing wall entity.
4. Add Property set and associated properties to an existing wall entity within the building model.

3.3 Use Case Diagram

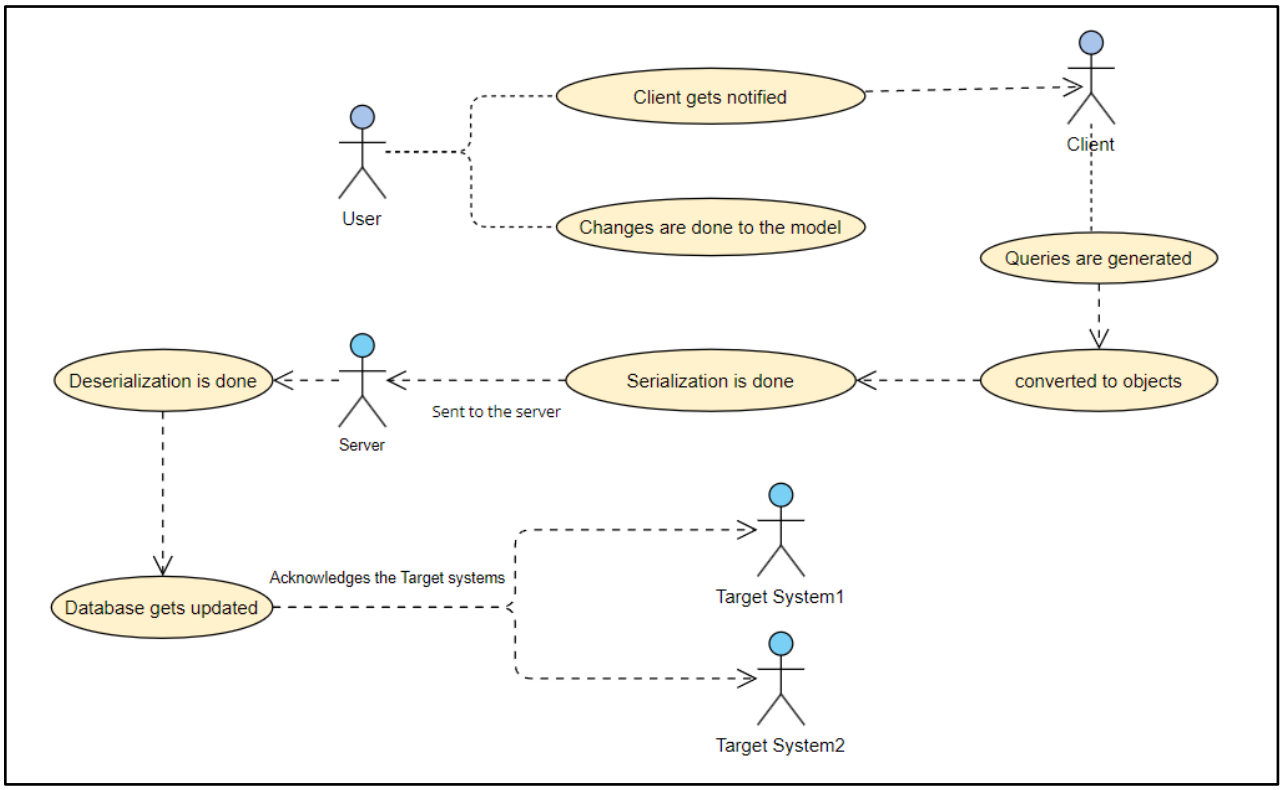

Figure 4 Use Case Diagram

A user interacts with the model within the Unity Environment, making changes or updates like moving an entity, changing the name of the entity, deleting an entity, or adding a new entity.

Upon making these changes, the client system is notified, and this notification triggers further actions. The client generates queries based on the changes made to the model and these queries are structured as objects. The queries generated are represented as objects and are then serialized.

Serialization is the process of converting these objects into a format that can be easily transmitted over the network. Serialized query requests are then sent to the server where the database is stored. Usually, this communication takes place over a network connection. After receiving the serialized queries in string format, the server interprets them. It then deserializes it, converting them back into a usable object format. The server system then performs the necessary database update actions in accordance with the content of the query object. This may involve adding, modifying, or deleting records in the database. Once the database is updated, the acknowledgement as well as the serialized

XML data is received by the target systems. This acknowledgment confirms that the changes made by the user in Unity have been successfully applied to the database.

3.4 Sequence Diagram

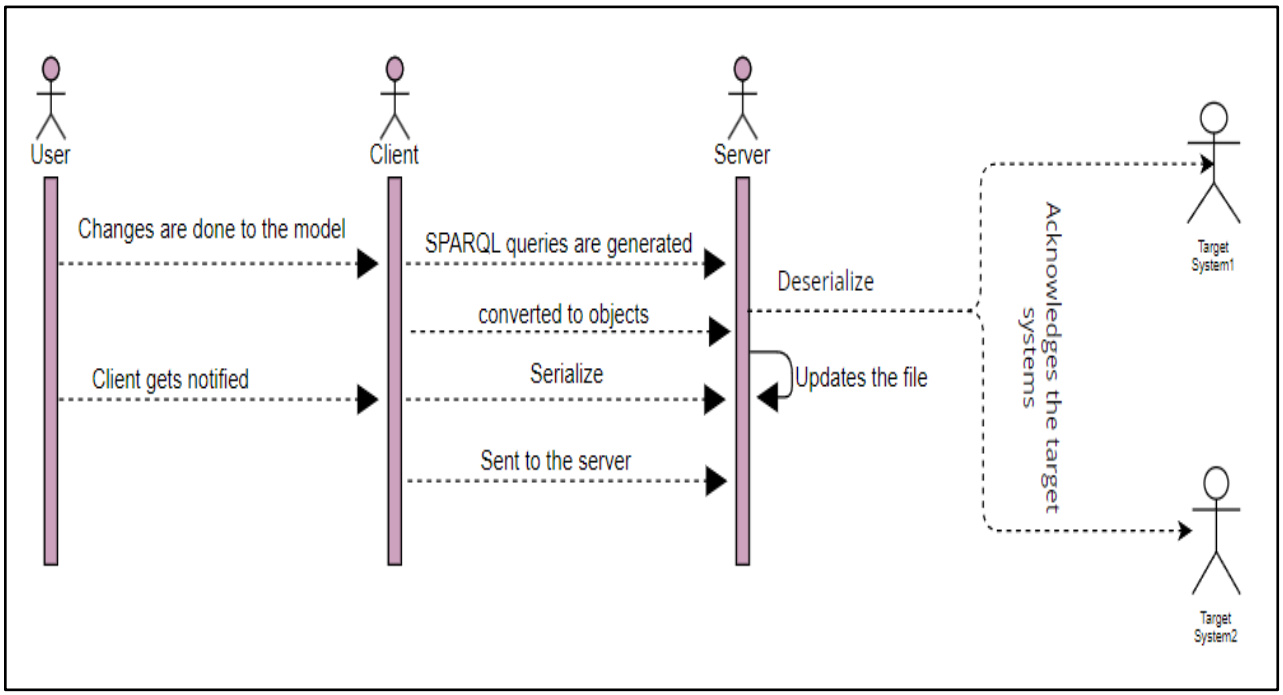

Figure 5 Sequence Diagram

The sequence begins with a user initiating some changes to the Unity model. Upon detecting changes in the Unity model, the client component is notified. The client component then generates queries based on the changes made to the Unity model. These queries are then converted to a suitable object format. It is then serialized and converted to bytes and sends the bytes over the network stream to the server system via network communication. The server then interprets the received response and then deserializes it to extract metadata and update query information. It updates the database based on the content of the object. The server then sends the received data from the client as well as the acknowledgment messages to specified target systems. The target systems process the acknowledgment, confirming the successful execution of the changes. This sequence diagram illustrates the seamless flow of events from user action to database update, ensuring that changes made in the Unity model are reflected accurately in the database. By employing serialization, along with the proper interpretation and processing on the server side, the system maintains reliability in handling user-initiated changes and database updates.

3.5 Client Architecture

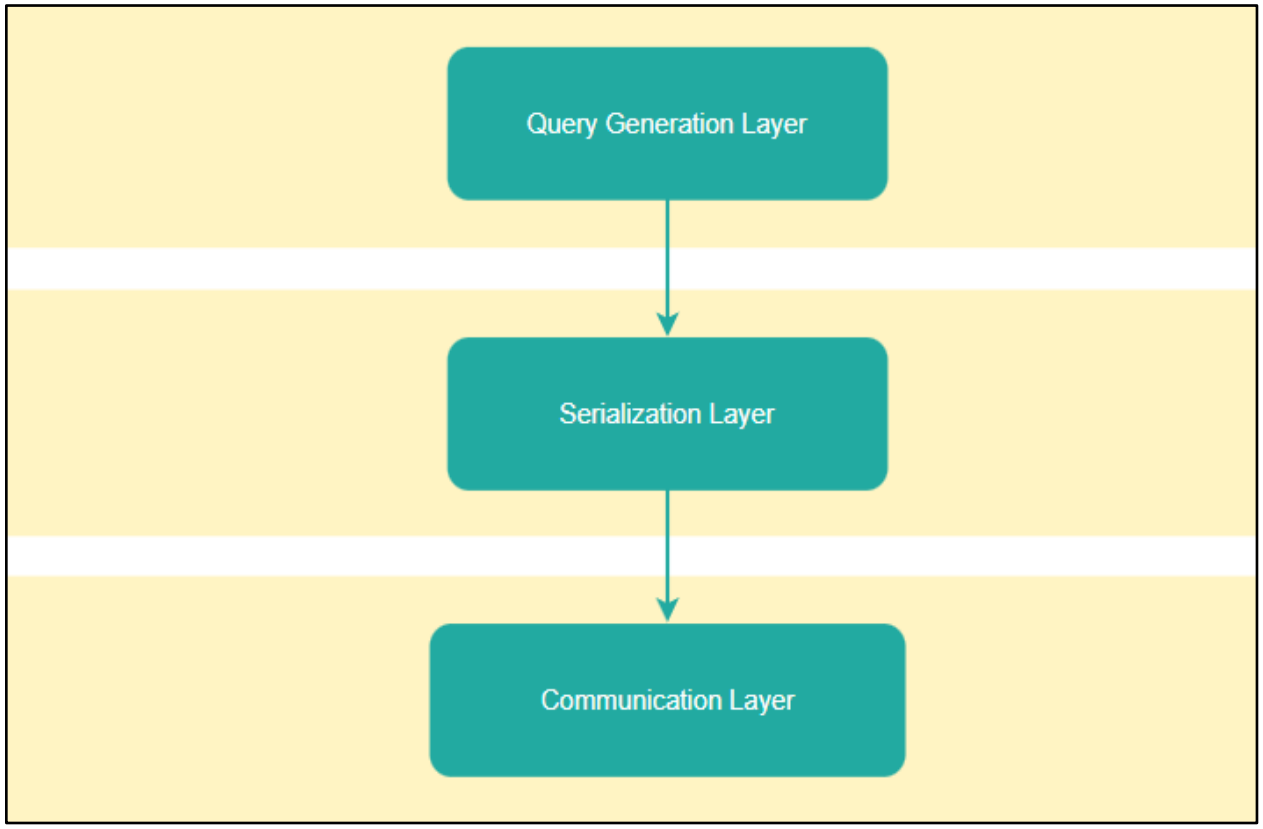

Figure 6 Client Architecture

- Query Generation Layer: This layer is responsible for constructing the queries based on the user interactions. It uses string interpolation to create queries that change based on the type of update needed. It also ensures that the queries conform to the correct syntax and standards.
- Serialization Layer: Serializes the queries to a suitable format for transmission. It also manages the metadata associated with queries, such as timestamps or relevant information during serialization.
- Communication Layer: This layer facilitates communication with the server over the network communication. It handles data transmission and reception, including establishing connections, sending data, and receiving responses.

3.6 Server Architecture

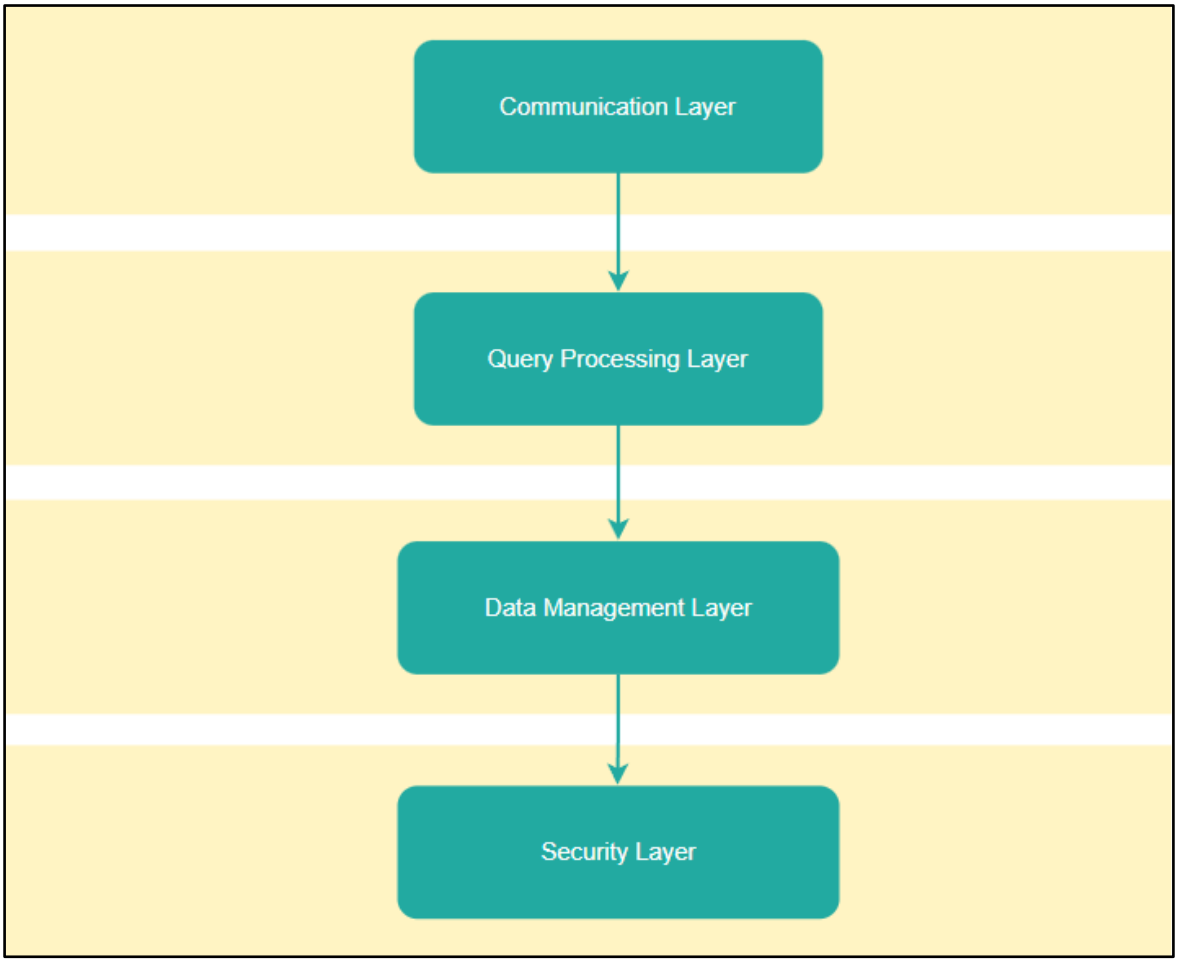

Figure 7 Server Architecture

- Communication Layer: Handles communication with clients over the network connection. It includes mechanisms for establishing, maintaining, and terminating connections. Managing data transfer, accepting client requests, and listening for new connections are some responsibilities of the communication layer.
- Query Processing Layer: This layer interprets the query received from the client. It then parses and validates the queries to ensure that they adhere to the query standard and the server's capabilities. Additionally, this layer may perform some transformations on these queries before executing them.
- Data Management Layer: Responsible for managing the database within the server. It handles operations such as updating and maintaining the data. This layer ensures consistency in storage and retrieval of the data.

- Security Layer: Handles authentication and authorization of clients to ensure only authorized users can access the server's resource.

This methodology provided insights into the following concepts, offering valuable ideas for the study.

Utilization of XBIM Library: The methodology utilizes the XBIM library to manipulate entities in the Revit model. This aligns with the findings from the literature review, which emphasized the importance of leveraging software libraries for BIM data manipulation and management.

SPARQL Queries for IFC File Modification: Drawing from the literature review on SPARQL query language, the methodology incorporates SPARQL queries to modify the IFC file. This approach aligns with the findings that SPARQL queries can be used effectively for BIM data extraction and modification.

Serialization and Deserialization: The methodology includes serialization and deserialization processes to facilitate data exchange and management between Revit models and IFC files. This aligns with the literature review, which highlighted the significance of serialization and deserialization for BIM data synchronization and exchange.

TCP Listener for Real-Time Communication: From the literature review on parallel programming and communication infrastructure, the methodology draws inspiration to introduce a TCP Listener as a key communication framework. This allows for real-time notification of changes made to the model and automatic generation of SPARQL queries, enabling efficient data exchange and collaboration.

By integrating these approaches identified in the literature review, the methodology provides a comprehensive solution for the utility of the XBIM library, the precision of SPARQL queries, and the robustness of serialization/deserialization techniques with a TCP Listener-based communication system.

4. IMPLEMENTATION

4.1 RDF Data:

RDF data is typically stored in files using various serialization formats such as Turtle, RDF/XML, N-Triples, etc. The FileLoader.Load method parses the RDF data file specified by the file Path parameter and extracts RDF triples from it. These triples represent statements in the form of subject-predicate-object, where the subject is a resource, the predicate is a property, and the object is either a resource or a literal value. As the RDF data is loaded, it is organized and stored in the Triple Store object, which acts as a repository for RDF triples. This allows for efficient storage and manipulation of RDF data within the program.

The methods responsible for updating RDF data in the program construct SPARQL Update queries made to specific operations, including renaming walls, adding new walls, managing owner history, and adding property sets and properties. These methods take parameters as necessary, such as wall identifiers and property information, and use string interpolation to dynamically generate the SPARQL update queries. Each method returns an `XElement` object representing the XML representation of the respective update query, which encapsulates the update operation and its metadata. These XML representations can then be serialized and transmitted to another system for execution, allowing for seamless integration and manipulation of RDF data across different systems.

4.2 SPARQL queries construction:

String interpolation, a feature in C#, is utilized to dynamically formulate SPARQL queries for updating RDF data within the program. This functionality enables the flawless inclusion of expressions or variables into string literals, allowing for the creation of customized queries. Within the SPARQL query template, placeholders enclosed in curly braces signify interpolation points where dynamic values are inserted. These values are derived from method parameters, variables, or constants within the program. During execution, the interpolation expressions are replaced with the actual values of the variables, resulting in a fully constructed SPARQL query string with dynamically integrated values. This method permits the development of queries made for specific update operations, including renaming walls, introducing new entities, managing owner history, and

defining property sets and properties. Through the utilization of string interpolation, the program achieves dynamic query generation, enhancing its versatility and efficacy in manipulating RDF data.

- The provided SPARQL query is formulated to insert data into an RDF graph, specifically targeting the representation of a new wall name within a building information model.

```
string updateString = $@"PREFIX ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC4/ADD1/OWL#>
                         PREFIX inst: <http://linkedbuildingdata.net/ifc/resources20200624_184152/>
                         PREFIX express: <https://w3id.org/express#>
                         PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                         INSERT DATA {{
                             inst:NewWall rdf:type ifc:IfcWall ;
                                          ifc:name_IfcRoot inst:IfcLabel_1337 ;
                                          ifc:globalId_IfcRoot inst:NewWallId ;
                                          express:hasString ""{wallName}"" .
                         }}";
```

Figure 8 Sparql Query for UpdateWallName

- The provided SPARQL query is structured to insert data into an RDF graph, particularly to define properties of a new wall ID within a building information model.

```
string updateString = $@"PREFIX ifc: <http://standards.buildingsmart.org/IFC/DEV/IFC4/ADD1/OWL#>
                         PREFIX inst: <http://linkedbuildingdata.net/ifc/resources20200624_184152/>
                         PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                         INSERT DATA {{
                             inst:{newWallId} rdf:type ifc:IfcWall ;
                                              ifc:name_IfcRoot inst:IfcLabel_{Guid.NewGuid()} ;
                                              ifc:globalId_IfcRoot inst:{newWallId} .
                         }}";
```

Figure 9 Sparql Query for AddNewWall

- The provided SPARQL query is designed to insert data into an RDF graph, specifically focusing on the representation of ownership history within a building information model.

```
string updateString = @"PREFIX ifc:  <http://standards.buildingsmart.org/ifc/dev/ifc4/add1/owl#>
                        PREFIX inst:  <http://linkedbuildingdata.net/ifc/resources/20240108_103400/>
                        PREFIX express:  <https://w3id.org/express#>
                        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                        INSERT DATA {
                        inst:ifcproject_100 ifc:ownerhistory_ifcroot inst:ifcownerhistory_new .
                        inst:ifcownerhistory_new a ifc:ifcownerhistory ;
                        ifc:owninguser_ifcownerhistory inst:ifcpersonandorganization_3 ;
                        ifc:owningapplication_ifcownerhistory inst:ifcapplication_1 ;
                        ifc:changeaction_ifcownerhistory ifc:added .
                        }";
```

Figure 10 Sparql Query for AddOwnerHistory

- The provided SPARQL query inserts data into an RDF graph, specifically targeting property sets and properties within a building information model.

```
string updateString = @"PREFIX ifc:  <http://standards.buildingsmart.org/ifc/dev/ifc4/add1/owl#>
                        PREFIX inst:  <http://linkedbuildingdata.net/ifc/resources20200624_184152/>
                        PREFIX express:  <https://w3id.org/express#>
                        PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                        PREFIX pset: <http://www.buildingsmart-tech.org/ifcowl/ifc4/psd#>

                        INSERT DATA {
                          inst:yourentity ifc:haspropertysets inst:yourpropertyset .
                          inst:yourpropertyset rdf:type ifc:ifcpropertyset ;
                                               ifc:globalid_ifcroot inst:yourpropertysetid ;
                                               ifc:ownerhistory_ifcroot inst:yourownerhistory ;
                                               ifc:name_ifcroot ""yourpropertyset"" ;
                                               ifc:description_ifcproperty ""yourpropertysetdescription"" ;
                                               ifc:hasproperties inst:yourproperty .
                          inst:yourproperty rdf:type ifc:ifcpropertysinglevalue ;
                                            ifc:globalid_ifcroot inst:yourpropertyid ;
                                            ifc:ownerhistory_ifcroot inst:yourownerhistory ;
                                            ifc:name_ifcroot ""yourpropertyname"" ;
                                            ifc:description_ifcproperty ""yourpropertydescription"" ;
                                            ifc:nominalvalue_ifcproperty ""yournominalvalue"" .
                    }";
```

Figure 11 Sparql Query for AddPropertySetAndProperty

4.3 Serialization of update queries:

Serialization of queries is a crucial step in communication between different systems. Update queries are generated to modify the RDF data according to specific requirements. These queries are expressed in SPARQL, a standardized query language for RDF data. Serialization refers to the process of converting these queries, along with the associated metadata, into a structured format that can be transmitted over the network. The serialized representation chosen for our system is XML, a widely used format for representing structured data.

The overview of XML serialization of update queries in the program involves the conversion of constructed SPARQL update queries, represented as XElement objects, into string format. Initially, these queries are constructed and encapsulated within XElement objects, which capture both the query content and any associated metadata, such as timestamps or relevant information. The serialization process then converts these XElement objects into strings, preserving the structure and content of the SPARQL Update queries in a standardized manner suitable for storage or transmission over the network. This transformation ensures that the resulting XML representation reflects the structure of the queries, with elements and attributes corresponding to different components of the queries.

4.3 Communication:

The `SendSerializedXmlToSystem2 method serves the important role of transmitting serialized XML data over a TCP/IP connection to another system, designated as a RDF Processing System. Initially, the method establishes communication with the RDF Processing System by specifying its IP address and port number, determining the destination for the data transmission. Prior to sending, the XML data representing SPARQL Update queries is serialized into a string format using UTF-8 encoding, converting the structured XML into a byte sequence suitable for network transmission. Subsequently, a network stream is obtained from the `TcpClient` object, serving as the conduit for transmitting data across the TCP connection. The serialized XML data, now represented as a byte array, is written to the network stream using the `Write` method, effectively sending the data to the RDF Processing System. Finally, upon completion of data transmission, the TCP connection is gracefully closed using the `Close` method to release network resources and terminate the communication session. Thus the `SendSerializedXmlToSystem2` method facilitates seamless inter-

system communication and data exchange by encapsulating the necessary steps for establishing a connection, serializing the data, and transmitting it over the network to the RDF Processing System.

4.4 Client-Server description:

The client component, encapsulated within the Client class, serves as the initiator of program execution. It fulfills several key responsibilities, including loading RDF data from a specified file path, constructing SPARQL update queries to manipulate the RDF data based on defined operations, serializing these queries into XML format, and finally transmitting them over a TCP/IP connection to another system, referred to as RDF Processing System or Server system. Its role would involve receiving the serialized XML data sent by the client over the TCP/IP connection, deserializing this data to extract the SPARQL update queries, and subsequently executing these queries to perform the intended operations on the RDF data stored within the RDF Processing System. Additionally, the server component would handle communication protocols and potentially enforce security measures to ensure the secure reception and processing of incoming data from clients. Overall, the client-server architecture facilitates the consistent exchange and manipulation of RDF data between the client and RDF Processing Systems, enabling effective collaboration and data management.

4.5 System Design:

4.5.1 Query Generation System (System 1):

Imports:

In programming, imports are used to bring in functionalities from external libraries or modules. For networking and IO operations in this case, libraries or modules for handling TCP connections, streams, and exceptions are typically imported.

TcpServer class:

This class serves as the core component of the server implementation. It encapsulates the server's functionality, including starting and stopping the server, handling client connections, and processing incoming data.

Main method:

The Main method is the entry point of the program. It initializes the server by creating a TcpListener object, which listens for incoming TCP connections on a specific IP address and port. Once the listener is started, it enters a loop to continuously accept incoming client connections. For each accepted connection, it typically spawns a new thread or uses an asynchronous method to handle the client separately, allowing the server to handle multiple clients simultaneously.

HandleClient method:

This method is responsible for handling each client connection. It receives a TcpClient object representing the connected client. The method retrieves the Network Stream associated with the client, which allows reading from and writing to the network. It then reads data from the network stream in a loop until there is no more data to read. The received bytes are typically converted into a human-readable format. Once the data is processed, the method closes the connection with the client.

Exception handling:

Exception handling is crucial for robustness and error management in server applications. It ensures that the server gracefully handles unexpected situations, such as network errors or client disconnects. By catching exceptions, the server can log errors and continue running without crashing.

The UML class diagram for this system would include classes such as Client and TripleStore, representing the client application and the collection of RDF triples, respectively.

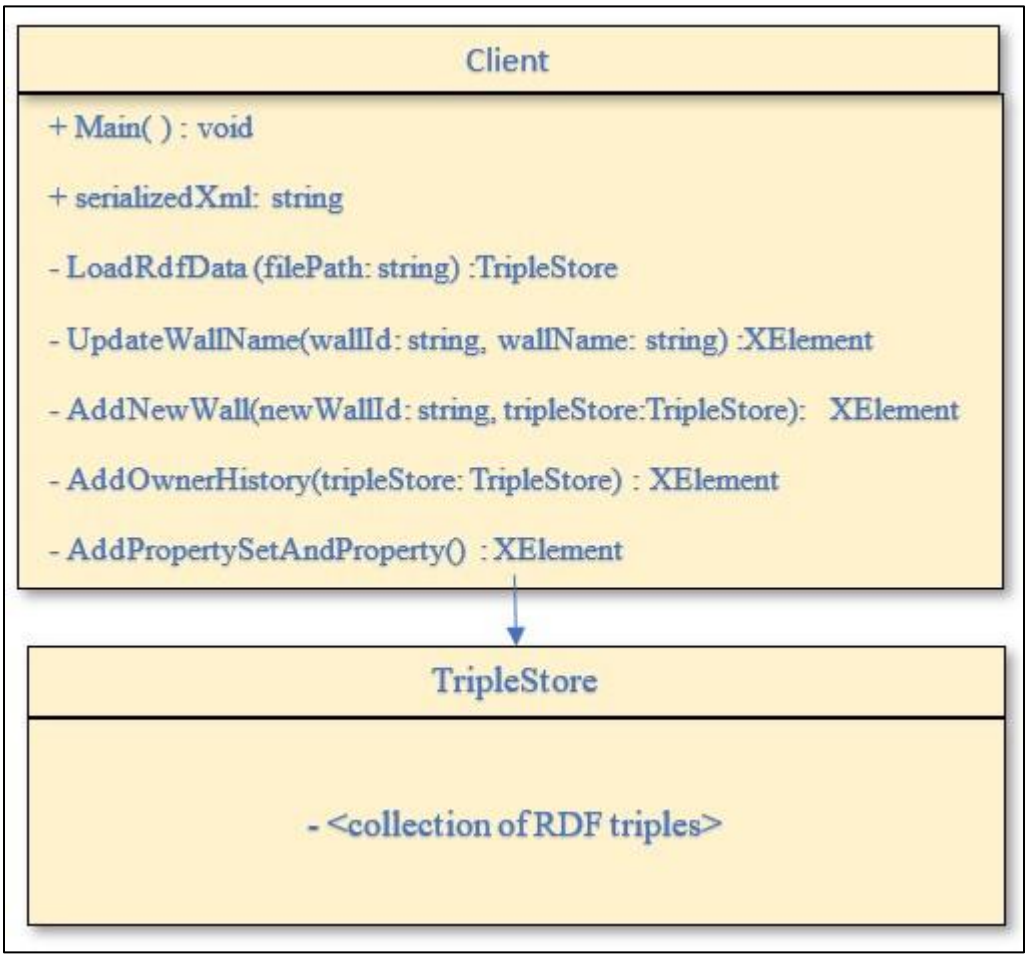

Figure 12 UML diagram for Query Generation System (System 1)

4.5.2 Data Processing System (System 2):

Imports:

The code imports various namespaces required for different functionalities such as Networking, for handling TCP connections (System.Net.Sockets).IO operations, for reading and writing files (System.IO). Threading is for handling multiple client connections concurrently (System.Threading). XML serialization is for converting XML data to and from objects (System.Xml.Serialization).RDF manipulation is for working with RDF data (Unknown namespaces)

Classes:

Target System represents details of a target system such as its name, IP address, and port. Messages represent a message containing serialized XML data and an acknowledgment message. Server contains the Main method and helper methods to manage client connections and update the RDF database.

Main method:

Creates a TcpListener instance to listen for incoming connections on a specified IP address and port. Accepts client connections in a loop, each in a separate thread to handle multiple clients concurrently.

HandleClient method:

Receives serialized XML data from the client. Deserializes the XML to extract metadata and update queries. Updates the RDF database using the received update queries. Saves the updated RDF data to a file. Sends serialized XML data and an acknowledgment message to multiple target systems.

Data Classes:

Update Query Metadata represents metadata and update queries extracted from the serialized XML. My Metadata contains timestamp and author information for the update. Update contains the SPARQL update query.

This UML diagram primarily consists of a class diagram illustrating the structure of the system. It includes classes such as Server as the main server handling client connections, Target System representing systems to be communicated, and Message for encapsulating communication data.

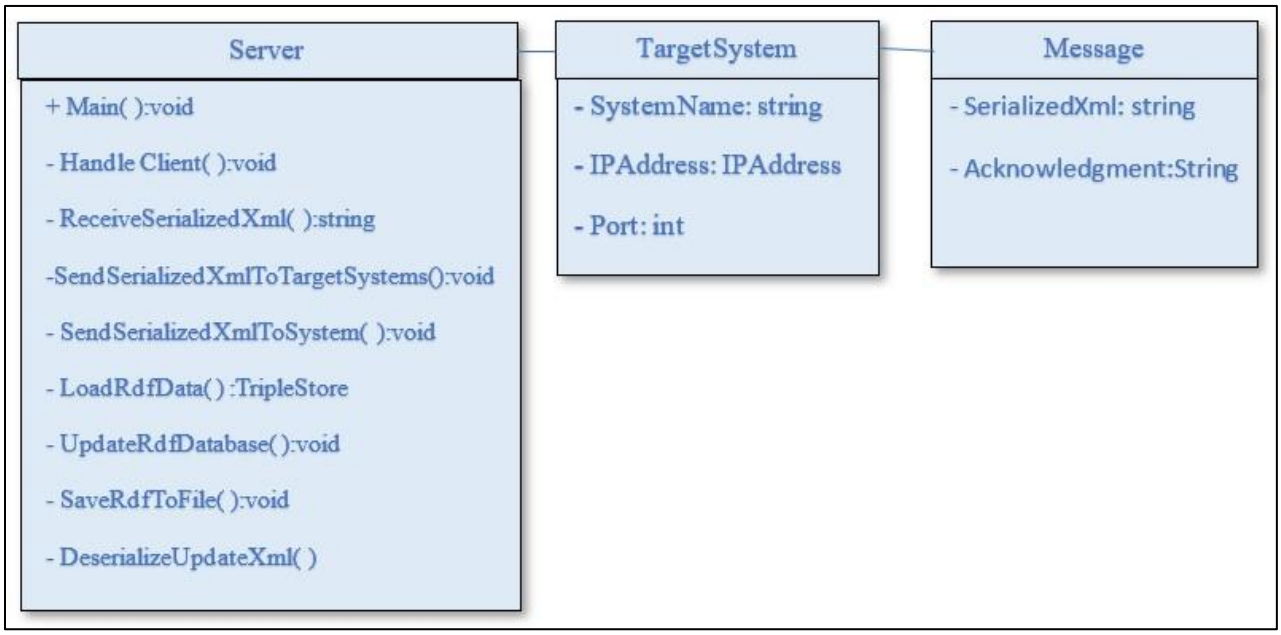

Figure 13 UML diagram for Data Processing System (System 2)

4.5.3 Target System (System 3):

Imports:

The code imports necessary namespaces such as **System.Net** and **System.Net.Sockets** for networking operations. **System.Text** is for handling character encodings.

TcpServer class:

It serves as the main class encapsulating server functionality. It contains the Main method, the entry point of the application. Provides a method named HandleClient to manage individual client connections.

Main method:

Creates a TcpListener instance to listen for incoming TCP connections on a specified IP address and port. Starts the TcpListener. Enter a loop to continuously listen for incoming client connections. Upon the client connection, it accepts the connection and typically spawns a new thread (or handles asynchronously) to manage that client.

HandleClient method:

Takes a TcpClient object representing a connected client. Retrieves the NetworkStream from the client, enabling reading and writing data. Reads data from the network stream in a loop until there's no more data to read. It converts the received bytes into a UTF-8 encoded string using a StringBuilder for efficient string manipulation. Prints the received string on the console and closes the TcpClient connection after handling the client.

Exception handling:

It catches any exceptions that may occur during the execution of the server or when handling a client connection. Error messages are printed to the console for debugging and error reporting purposes.

The UML diagram consists of a class diagram illustrating the structure of the system. In this diagram, there is one class, TcpServer, representing the main server application. Inside this class, there would be two main methods: Main () and Handle Client ().

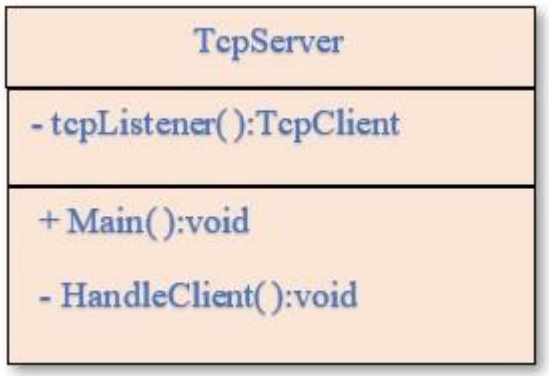

Figure 14 UML diagram for Target System(System 3)

5. TESTING

Throughout the project, several challenges were encountered, each requiring specific investigative and problem-solving approaches to overcome.

5.1 Initial Attempt with BIM Server:

The attempt to utilize the BIM Server initially fell short of expectations, as it did not effectively fulfill the intended purpose.

5.2 Trouble with Apache Jena Server and Updating IFC File:

Difficulties arose when attempting to update the IFC file using the Apache Jena Server. The complexity of configuring the server to handle SPARQL queries and perform the necessary conversions from IFC to RDF format was initially underestimated.

5.3 Discovering the Need to Convert IFC to RDF:

Recognizing the complexities involved in updating the IFC file, it became apparent that converting it to RDF format could offer greater flexibility. Converting IFC files to RDF format was a critical step in enabling SPARQL query manipulation but proved to be a complex task due to the detailed nature of BIM data and the need for precise data representation.

5.4 Challenges with SPARQL Query Generation:

Generating SPARQL queries to accurately reflect the desired modifications to the RDF data was initially challenging, requiring thorough exploration of SPARQL documentation for guidance and understanding. The queries needed to be precise and efficient to ensure the correct manipulation of the BIM data within the RDF framework. A detailed understanding of SPARQL syntax and its application within the context of BIM data was required. Additionally leveraging resources such as online forums, SPARQL documentation, and examples helped in crafting more effective queries.

5.5 Difficulty in Deserializing XML Object and Updating Turtle File:

Difficulties were encountered in the process of deserializing XML objects and updating turtle files (RDF format), particularly in managing data formats and transformations.

6. CONCLUSION AND DISCUSSIONS

The primary aim of this project was to enable real-time collaboration and synchronization of data between Revit models and IFC files, addressing a critical gap in Building Information Modeling processes. To achieve this, the project leveraged a multi-faceted technological approach that included the utilization of the XBIM library, SPARQL queries for data manipulation, and a comprehensive system for serialization and deserialization. Additionally, a TCP Listener-based communication infrastructure was established to support real-time data exchange and notifications between different system components.

The solution's architecture was designed to integrate into existing BIM workflows, thus enhancing efficiency and collaboration in architectural design and construction projects. At its core, the architecture utilized the XBIM library to enable direct manipulation of entities within Revit models, facilitating actions such as the addition, deletion, or updating of entities like walls. To address the challenge of updating IFC files to reflect changes made in the Revit model, the project employed SPARQL queries. SPARQL emerges as a pivotal tool for querying and manipulating data stored in RDF format. RDF, with its data model representing information in subject-predicate-object triples, aptly suits the representation of diverse and interconnected data, especially within domains like the Semantic Web and building information modeling. These queries allowed for modification of the IFC files by converting them into RDF format, which could then be easily manipulated and updated.

Serialization played a critical role in preparing these SPARQL queries for network transmission, converting them into an XML format suitable for both storage and communication. Deserialization, on the other hand, enabled the conversion of received data back to its original form, ensuring that both Revit models and IFC files remained synchronized.

TCP is a fundamental protocol in computer networking that ensures reliable and connection-oriented communication between devices over IP-based networks. Its key role is to guarantee the integrity of data transmission by employing various mechanisms such as acknowledgments, retransmissions, and sequencing. This ensures that data sent from one endpoint is received correctly and orderly by the destination endpoint. The introduction of a TCP Listener as the communication system allowed for the real-time transmission of serialized data between the client and the server. This setup ensured that any modifications made to the Revit model were reflected in the RDF files, thereby maintaining

consistency across BIM documents and facilitating a smoother collaborative process among project participants. Moreover, TCP plays a pivotal role in enabling bidirectional communication between two critical systems: the Query Generation System (System 1) and the Data Processing System (System 2). Through TCP clients and servers integrated into the codebase, these systems establish a reliable communication channel. Serialized XML data containing updates to the Resource Description Framework (RDF) database is transmitted from the Query Generation System to the RDF Processing System via TCP, facilitating efficient data exchange and synchronization between the two systems.

Overall, the solution presented a robust and innovative approach to overcoming the traditional barriers to real-time BIM collaboration, setting a new standard for efficiency and effectiveness in the management and synchronization of BIM data.

6.1 Key Findings

The project's exploration into enhancing Building Information Modeling workflows has yielded significant findings, notably in the realm of automating updates to IFC files, a critical component in the collaborative construction and design process. A standout achievement was the successful implementation of a system where modifications made to a Revit model automatically triggered updates in the corresponding IFC files. This automation was largely facilitated by the innovative use of SPARQL queries stored in the RDF format. By converting IFC files into this format, the project enabled a more flexible and dynamic approach to managing BIM data, allowing for real-time updates and alterations.

This process was underpinned by a comprehensive communication infrastructure, utilizing a TCP Listener to facilitate the exchange of data between client and server systems. This system ensured that changes made to the Revit model were immediately propagated to the RDF files, thereby maintaining a consistent and up-to-date representation of the project across all platforms and participants. The synchronization achieved between the client and server systems represents a major leap forward in BIM technology, eliminating many of the manual processes traditionally associated with updating and managing IFC files.

Moreover, the project's communication framework was not only effective but also robust, capable of handling the complex data exchanges required for real-time BIM collaboration.

6.2 Technological and Methodological Insights

The project's technological and methodological choices were pivotal in addressing the requirements of real-time data management and collaboration within the Building Information Modeling ecosystem. The selection of the XBIM library, dotNet rdf library, RDF, and SPARQL was strategic, directly influencing the project's success.

6.2.1 Technological Choices:

- XBIM Library: The XBIM library was instrumental in manipulating BIM data within Revit models. Its robust API facilitated the introduction, modification, and deletion of entities like walls, effectively bridging the gap between the conceptual design and the data model representation. This choice underscored the project's commitment to leveraging open-source tools for enhancing BIM workflows, ensuring compatibility and ease of integration with existing systems.
- dotNet RDF Library: Utilized for handling RDF data and SPARQL queries, the dotNet rdf library provided a solid foundation for managing complex data structures inherent to BIM data. Its ability to seamlessly serialize and deserialize RDF data ensured that modifications to the Revit model were accurately reflected in the IFC files, enhancing the reliability of data exchanges.

6.2.2 Methodological Approaches:

- Using RDF and SPARQL: The adoption of RDF for representing IFC files and SPARQL for querying and updating this data was a methodological cornerstone. This approach enabled a granular level of control over BIM data, allowing for precise modifications and updates. The flexibility and expressive power of SPARQL queries facilitated complex data manipulations, demonstrating the effectiveness of semantic web technologies in the context of BIM.

7. SUMMARY

"Real-Time Collaboration on Building Models" presents a robust framework aimed at improving communication and collaboration within architectural design and building information modeling (BIM) processes. The project introduces a novel approach utilizing serialized XML objects to facilitate real-time updates to Resource Description Framework (RDF) data, crucial for structuring information in the construction industry. Leveraging Industry Foundation Classes (IFC), the project ensures standardized BIM data exchange, complemented by the use of queries to accurately reflect modifications in BIM models. Serialization and deserialization processes are integrated to enhance data exchange efficiency. A robust communication infrastructure supports real-time collaboration among stakeholders, with notifications enabling prompt updates. Implementation details cover RDF data handling, SPARQL query construction, serialization of update queries into XML format, and TCP communication for data transmission. The client-server architecture enables seamless exchange and manipulation of RDF data between systems. The project's success lies in its ability to enable real-time collaboration and synchronization between Revit models and IFC files, bridging gaps in BIM workflows and ensuring efficiency and accuracy. Key findings highlight achievements in automating updates to IFC files through innovative SPARQL queries and robust communication infrastructure. Technological choices, such as the XBIM and dotNet RDF libraries, underscore the project's effectiveness in enhancing BIM workflows. Overall, "Real-Time Collaboration on Building Models" offers a comprehensive solution to industry challenges, enhancing efficiency and accuracy in architectural design and construction processes through real-time collaboration and synchronization mechanisms.

8. REFERENCES

[1] D. Guo, E. Onstein, and A. D. L. Rosa, "A Semantic Approach for Automated Rule Compliance Checking in Construction Industry," *IEEE Access*, vol. 9, pp. 129648–129660, 2021, doi: 10.1109/ACCESS.2021.3108226.

[2] Z. Wang, R. Sacks, B. Ouyang, H. Ying, and A. Borrmann, "A Framework for Generic Semantic Enrichment of BIM Models," *J. Comput. Civ. Eng.*, vol. 38, no. 1, p. 04023038, Jan. 2024, doi: 10.1061/JCCEE5.CPENG-5487.

[3] M. Das, J. C. P. Cheng, and S. Shiv Kumar, "BIMCloud: A Distributed Cloud-Based Social BIM Framework for Project Collaboration," in *Computing in Civil and Building Engineering (2014)*, Orlando, Florida, United States: American Society of Civil Engineers, Jun. 2014, pp. 41–48. doi: 10.1061/9780784413616.006.

[4] W. Mazairac and J. Beetz, "BIMQL – An open query language for building information models," *Advanced Engineering Informatics*, vol. 27, no. 4, pp. 444–456, Oct. 2013, doi: 10.1016/j.aei.2013.06.001.

[5] D. Guo, E. Onstein, and A. D. L. Rosa, "An Approach of Automatic SPARQL Generation for BIM Data Extraction," *Applied Sciences*, vol. 10, no. 24, p. 8794, Dec. 2020, doi: 10.3390/app10248794.

[6] A. Kandil and K. El-Rayes, "Parallel Genetic Algorithms for Optimizing Resource Utilization in Large-Scale Construction Projects," *J. Constr. Eng. Manage.*, vol. 132, no. 5, pp. 491–498, May 2006, doi: 10.1061/(ASCE)0733-9364(2006)132:5(491).

[7] E. Alreshidi, M. Mourshed, and Y. Rezgui, "Cloud-Based BIM Governance Platform Requirements and Specifications: Software Engineering Approach Using BPMN and UML," *J. Comput. Civ. Eng.*, vol. 30, no. 4, p. 04015063, Jul. 2016, doi: 10.1061/(ASCE)CP.1943-5487.0000539.

[8] M. Yin *et al.*, "Two-stage Text-to-BIMQL semantic parsing for building information model extraction using graph neural networks," *Automation in Construction*, vol. 152, p. 104902, Aug. 2023, doi: 10.1016/j.autcon.2023.104902.

[9] N. Wang, R. R. A. Issa, and C. J. Anumba, "NLP-Based Query-Answering System for Information Extraction from Building Information Models," *J. Comput. Civ. Eng.*, vol. 36, no. 3, p. 04022004, May 2022, doi: 10.1061/(ASCE)CP.1943-5487.0001019.

[10] G. Imre, M. Kaszó, T. Levendovszky, and H. Charaf, "A Novel Cost Model of XML Serialization," *Electronic Notes in Theoretical Computer Science*, vol. 261, pp. 147–162, Feb. 2010, doi: 10.1016/j.entcs.2010.01.010.

[11] L. Zhang and R. R. A. Issa, "Ontology-Based Partial Building Information Model Extraction," *J. Comput. Civ. Eng.*, vol. 27, no. 6, pp. 576–584, Nov. 2013, doi: 10.1061/(ASCE)CP.1943-5487.0000277.

[12] J. Werbrouck, O. Schulz, J. Oraskari, E. Mannens, P. Pauwels, and J. Beetz, "A generic framework for federated CDEs applied to Issue Management," *Advanced Engineering Informatics*, vol. 58, p. 102136, Oct. 2023, doi: 10.1016/j.aei.2023.102136.

[13] X. Tao, M. Das, Y. Liu, and J. C. P. Cheng, "Distributed common data environment using blockchain and Interplanetary File System for secure BIM-based collaborative design," *Automation in Construction*, vol. 130, p. 103851, Oct. 2021, doi: 10.1016/j.autcon.2021.103851.

[14] S. Esser, S. Vilgertshofer, and A. Borrmann, "Version control for asynchronous BIM collaboration: Model merging through graph analysis and transformation," *Automation in Construction*, vol. 155, p. 105063, Nov. 2023, doi: 10.1016/j.autcon.2023.105063.

[15] https://de.wikipedia.org/wiki/Building_Information_Modeling

[16] https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/

[17] https://www.w3.org/TR/sparql11-query/

[18] https://dotnetrdf.org/docs/stable/user_guide/getting_started.html