IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 28 June 2026, accepted 17 July 2026, date of publication 4 August 2026, date of current version 17 August 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3720274

RESEARCH ARTICLE

BIM-GRASP: A Graph-RAG Approach for Semantic Parsing of IFC Building Models

HADEEL SAADANY[1], SIMA IRANMANESH[1], MALIK UZAIR MEHMOOD[1], EDLIRA VAKAJ[1], AND EDDIE KASSAEI[2]

[1]School of Computing, Birmingham City University, B47RQ Birmingham, U.K.
[2]Inertia Systems, Chula Vista, CA 91915, USA

Corresponding author: Hadeel Saadany (Hadeel.Saadany@bcu.ac.uk)

This work was supported in part by the European Union's Horizon Europe Research and Innovation Program through Automated Compliance Checks for Construction, Renovation or Demolition Works (ACCORD) under Grant 101056973; and in part by U.K. Participants in Horizon Europe Project (ACCORD) funded by UK Research and Innovation (UKRI) through Cardiff University, Birmingham City University, and Building Smart International under Grant 10040207, Grant 10038999, and Grant 10049977.

ABSTRACT Building Information Modeling (BIM) is central to modern construction and design, with the Industry Foundation Classes (IFC) format serving as a widely adopted open standard for representing building models. However, IFC data is notoriously complex as it encodes structural, geometric, and semantic information in deeply nested relationships that are difficult to interpret without specialized tools and domain expertise. This complexity limits access to meaningful insights, particularly for non-technical stakeholders. We introduce BIM-GRASP, a novel system-level application that combines Graph-Retrieval Augmented Generation (Graph-RAG) with Generative AI to enable natural language interaction with IFC building models. BIM-GRASP transforms IFC files into a knowledge graph, which is then queried by a Large Language Model (LLM) to answer questions about building elements such as geometric attributes, material specifications, and inter-element relationships. Unlike traditional IFC parsers, BIM-GRASP supports complex queries that require reasoning across multiple layers of the IFC hierarchy as it is able to retrieve information from disparate parts of the model. This allows users to ask questions and receive accurate, context-aware answers in natural language, without relying on schema-level navigation or technical syntax. Our findings show that providing the model with domain-specific In-Context Learning (ICL) significantly improves precision and relevance in information extraction, achieving an average accuracy of 88% across diverse building models and query types. To the best of the authors' knowledge, BIM-GRASP is the first framework of its kind to enable the parsing of IFC data as a knowledge graph through natural language without requiring specialized technical expertise or manual schema navigation. By bridging the gap between complex building data and non-technical stakeholders, BIM-GRASP accelerates decision-making and improves transparency across the built environment.

INDEX TERMS Building information modeling (BIM), generative AI application, graph-RAG, industry foundation classes (IFC), semantic querying.

I. INTRODUCTION

BIM (Building Information Modelling) is widely applied in the Architecture, Engineering, Construction, and Operation (AECO) sectors to organize and disseminate building-related data throughout all phases of a construction project. Since the introduction of the IFC (Industry Foundation Classes) standard by buildingSMART [1], most BIM platforms have adopted IFC as the primary format for storing, exchanging, and retrieving information about architectural components [2], [3]. IFC files are data-dense and structurally intricate text-based formats designed to enable data sharing across disciplines and software platforms throughout the project life cycle. While several commercial and open-source tools support interaction with IFC data [4], [5], [6], retrieving useful information from these files often demands not only

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Wang.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

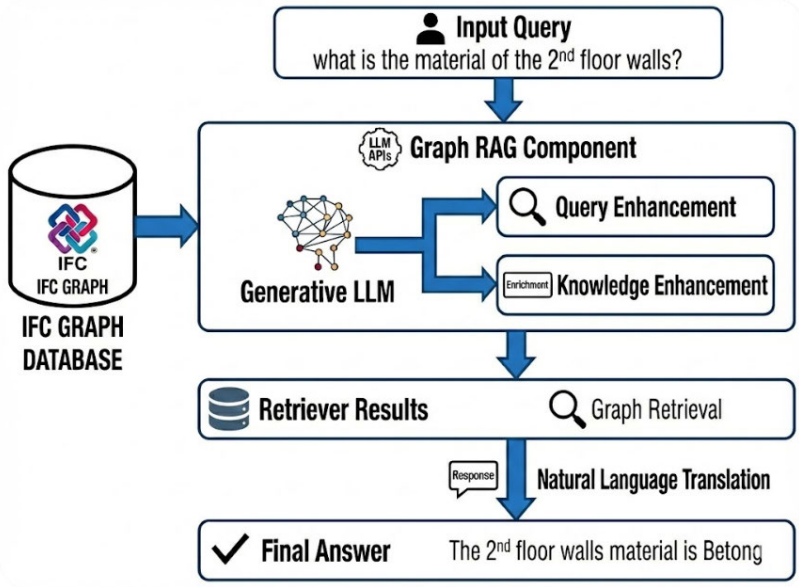


FIGURE 1. BIM-GRASP retrieval system queries a generative language model, by asking questions in natural language about specific information in the IFC data. We transform the IFC data into a knowledge graph. The LLM uses: 1) its pretrained knowledge, 2) the knowledge graph, and 3) the query to extract the correct answer from the complex IFC data.

technical familiarity with the intricate structure of the IFC schema but also practical experience with specialized BIM software tools. Additionally, many open-source IFC parsers may compromise semantic integrity when comparing or merging entities across IFC files [7].

To address these difficulties, numerous research efforts have introduced automated tools and methods aimed at simplifying the parsing of complex IFC data. These approaches typically involve converting IFC content into alternative formats using rule-based mappings [8], graph-theoretical models [9], structured query languages [10], or ontology-based representations [11]. However, such solutions often require detailed expertise in query syntax and mapping logic to extract specific attributes or entity relationships. Furthermore, if these rules are incomplete or ambiguous, the results may also be partial or not aligned with user expectations [12].

More recently, research has explored the application of Natural Language Processing (NLP) techniques for enabling more intuitive access to IFC data. NLP-driven methods for information retrieval have employed syntactic analysis [13], dependency-based parsing [14], and keyword classification approaches [15]. While the body of work on natural language interfaces for IFC data remains limited, these methods have shown considerable promise in improving access to building information models by simplifying the interaction with IFC complex data [16], [17].

In this study, we enhance NLP strategies by utilizing the capabilities of Generative AI in conjunction with Graph-RAG methodologies to construct a question-answering (QA) system designed to interpret IFC data, with both user queries and system outputs formulated in plain language. We demonstrate that Graph-RAG provides a highly suitable method for extracting and reasoning over IFC data for two key reasons. Firstly, the complex interconnections among components defined in IFC schemas such as walls, doors, floors, and their associated attributes exhibit inherently graph-like characteristics. In these graphs, nodes represent discrete entities like building components (e.g., *IfcWall*, *IfcDoor*) with descriptive attributes (e.g., *Thermal mass*, *GrossVolume*), while edges define their interrelations, including spatial hierarchies (*IfcRelContained*, *InSpatialStructure*) or physical connections (*IfcRelConnects*) [1]. Secondly, Graph-RAG enhances retrieval by returning cohesive subgraphs or groups of related entities rather than isolated data points. For instance, in response to a query such as "*Which wall has the highest dimension's volume in this floor?*", Graph-RAG identifies and retrieves a subgraph that encompasses the spatial configuration of the relative floor, the walls contained within it, and the volume attributed to those walls. This approach guarantees the extraction of data alongside the essential contextual relationships, thereby improving the precision and relevance of the system's answers.

The most critical feature of our parsing system is its ability to enable intuitive, natural language interaction with complex IFC building data, thus eliminating the need for users to possess technical expertise or familiarity with schema-level representations. Figure 1 represents the pipeline used for our BIM-GRASP retrieval system.[1] As illustrated in Figure 1, the BIM-GRASP multi-stage retrieval workflow initiates with an **Input Query** provided in natural language

[1] A video demonstration showcasing the real-time operation of the system is publicly accessible at: https://www.youtube.com/watch?v=F17NfOchfc8

(e.g., "*What is the material of the 2nd floor walls?*"). This query, alongside structured building data supplied by the **IFC Graph Database**, is fed into the core **Graph RAG Component**. Within this module, the system drives two parallel optimizations: 1) **Query Enhancement**, which utilizes relational data retrieved from the graph to refine the user's search intent, and 2) **Knowledge Enhancement**, which deploys the Generative LLM's foundational pre-trained knowledge to contextualize the query with broader domain awareness. Following these enhancements, the system executes a **Graph Retrieval** to isolate the pertinent data paths in the graph, yielding specific **Retriever Results**. In the final stage, the framework performs a **Natural Language Translation** on the retrieved graph data, converting complex relational insights into a user-friendly **Final Answer** (e.g., "*The 2nd floor walls material is Betong*"). The detailed methodology for each of these components is explained in Section IV.

Our top-performing language model utilizes In-Context Learning (ICL) with GPT-4.1, achieving an average accuracy of 88%. It is capable of accurately answering questions of progressively increasing complexity. These questions are derived from building models of varying scales, ranging from simple structures to multi-storey developments. These results demonstrate that our Graph-RAG framework, combined with specialized generative AI capabilities, effectively reduces the likelihood of hallucinations or interpretive errors when retrieving and reasoning over complex datasets such as those defined by the IFC schema. Moreover, due to the inherent intricacy of IFC data which often poses difficulties for individuals without technical expertise our BIM-GRASP retrieval system provides an intuitive and user-friendly interface by supporting natural language queries and easily understandable responses.

Thus, our contribution in this research are as follows:

1) We apply Graph-RAG to model IFC data as graphs, enabling structured retrieval and contextual reasoning.
2) We develop BIM-GRASP, a strong system-level application and natural language QA tool that interprets IFC queries and returns plain-language answers, adapting existing Graph-RAG methodologies specifically for the structural complexities of the IFC schema.
3) Our GPT-4.1 model, optimised with In-Context Learning (ICL), achieves 88% accuracy across diverse building models and query types.
4) We release our BIM-GRASP codebase and retrieval algorithm as open source, enabling replication and adaptation for other IFC use cases.

The remainder of this paper is structured as follows. Section II surveys key research efforts on parsing IFC data, alongside emerging work in NLP focused on generative language models. Section III provides a concise overview of the IFC schema and the Graph-RAG technique as preliminaries for addressing the challenge of transforming intricate IFC data into a structured knowledge graph for the BIM-GRASP system. Section IV elaborates our three-phase methodology for constructing graph representations and implementing Graph-RAG-based retrieval from IFC content. Section V presents the results and error analysis. Finally, Section VI concludes the paper by discussing the benefits, challenges, and potential future improvements of our proposed method. Both the IFC data and the source code supporting our LLM-enhanced BIM-GRASP system are publicly available at https://github.com/SimaIm/BIM-GRASP.

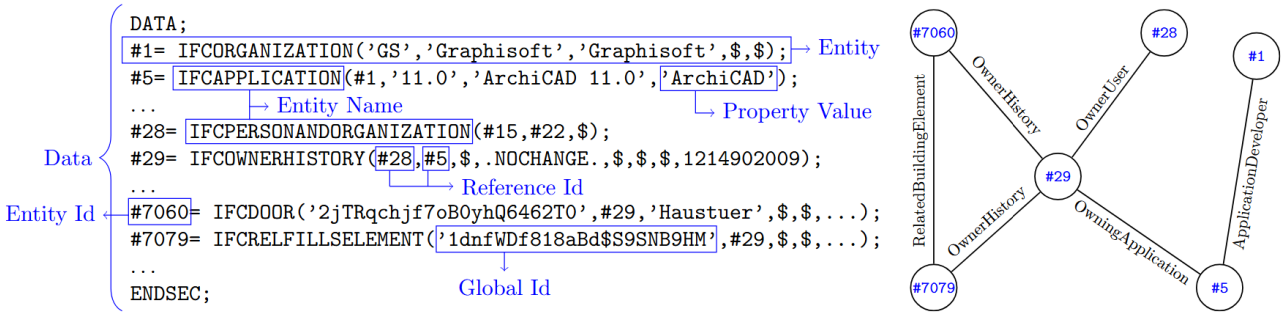
II. RELATED RESEARCH

Given the inherently complex nature of IFC models, numerous studies have investigated information extraction techniques to address diverse objectives, including verifying compliance with modelling standards [18], detecting differences between IFC files [19], and facilitating data retrieval, interpretability, and integration [12]. While research on IFC data extraction spans diverse goals, its primary focus has been on retrieving accurate and comprehensive information from the various elements embedded within building models. This is often achieved by transforming the IFC file into an alternative data representation, which then allows information retrieval through algorithmic methods [20], structured query languages such as SPARQL [21], or graph-theoretic and ontology-based frameworks [9].

However, the application of these methods remains challenging, as they require proficiency in specialised query languages (e.g., SPARQL) and a solid understanding of the target data format produced from the IFC model. To address the complexity associated with mapping-based IFC retrieval methods, recent research has increasingly explored the use of NLP techniques to facilitate information extraction from IFC data. An example is the work of [16] which proposed an intelligent data retrieval and representation framework for cloud-based BIM applications, employing NLP techniques to map user requirements to IFC entities and properties through keyword matching and constraint classification. Another example is [17], which introduced a three-stage QA system for IFC data retrieval. This system integrates a natural language query analyser, a content-word classifier, and a language generation module that converts retrieved BIM information into natural language responses.

Building on these examples, a common objective in most NLP-driven IFC retrieval research is to ensure both user queries and system's responses are expressed in plain human language [22]. However, such systems often depend on intricate NLP pipelines. In these workflows, IFC textual data may be processed for part-of-speech (POS) tagging and dependency parsing to identify relationships between entities [17]. The input sentence can be transformed into a syntactic tree representation to capture its grammatical structure [15], or a machine learning classifier may be employed to associate extracted regulatory concepts with their most relevant IFC concepts [23].

Only a limited number of studies have explored the use of generative LLMs for IFC data retrieval [24], [25]. For instance, [24] proposed a prompt-driven virtual assistant



(a) IFC (b) Graph

FIGURE 2. (a) A sample data section of an IFC file, and (b) A sample graph generated from the entities shown in the sample IFC file.

framework built on GPT 3.5, designed to facilitate NLP-based information extraction from BIM models. More recently, [25] introduced a proof-of-concept that applied Graph-RAG with LLMs to IFC data and demonstrated its feasibility on a small IFC model. However, the integration of generative AI and Graph-RAG for developing a comprehensive question–answering system capable of processing IFC data on a scale remains unexplored.

To understand the challenge of transforming intricate IFC data into a structured knowledge graph for the BIM-GRASP system, we provide a concise overview of the IFC schema and the Graph-RAG technique in the following sections.

III. PRELIMINARIES

Before detailing the BIM-GRASP methodology, it is essential to establish the foundational concepts that underly our framework. This section provides a concise overview of the three core elements utilized in our system. First, we outline the IFC schema to highlight the complexities of its deeply nested, object-oriented data structure. Second, we discuss the principles of Graph-RAG and explain how it addresses the reasoning gaps and structural limitations of standard vector-based RAG architectures. Finally, we introduce the Cypher query language, which serves as the primary mechanism for extracting structured relational patterns from our knowledge graphs. Together, these preliminaries illustrate the challenges of and solutions for transforming intricate IFC data into a format suitable for natural language parsing.

A. INDUSTRY FOUNDATION CLASSES

The Industry Foundation Classes (IFC) is a standardized, open data model developed by buildingSMART for representing building and construction data across different software platforms [26]. It is widely adopted in BIM workflows as a neutral format for exchanging complex building data that includes not only geometry, but also semantic information such as materials, classifications, and inter-element relationships [2], [3]. IFC files are hierarchical and object-oriented as it encodes a building's components and systems in deeply nested structures, which can make them challenging to parse without domain-specific tools and expertise.

An IFC file is composed of two main sections: 1) Header, which includes metadata such as the file's creation date and the schema version used, and 2) Data section, which contains the actual building model data. This data includes both geometric and semantic content, as well as connections between entities. Figure 2(a) showcases an example that demonstrates how an IFC file is structured. Each line begins with a unique identifier (eg #29), followed by the entity type (e.g. *IfcWall*), and associated attributes (e.g. '*Graphisoft*'). Links between elements are encoded using reference IDs (e.g. #28, #5). Many entities also carry a Global Unique Identifier (GUID), allowing for unique identification (e.g. Global Id shown in Figure 2(a)). Despite this, GUIDs can often change between file exports, which presents challenges in tracking elements across versions [3].

Understanding IFC data structures is a non-trivial task due to their deep hierarchical design. This difficulty arises not only from the dense and interconnected nature of relationships found in construction models, but also from the often non-descriptive property names and opaque reference structures within IFC entities (e.g. see the IFC43 annex for alphabetical listings of the IFC entities[2]). Consequently, individuals who are not familiar with the IFC schema may find it difficult to extract or interpret relevant details. For example, in Figure 2(a), someone without prior IFC experience may not realize that the string 'Haustuer' refers to the 'Name' attribute of the #7060, *IfcDoor* element. This entity is a specialization of the general class *IfcProduct*. According to the IFC schema [1], an *IfcDoor* can represent different roles. Also, as seen in Figure 2(a), this element in the building has a unique Global Id which is connected to element #29, that is the *IfcOwnerHistory*. The latter defines all history and identification related information to this door in the building as well as its relationships and properties. As can be seen from the IDs of elements in Figure 2(b), other entities are intricately connected with this element in the building. This complexity in handling the

[2]IFC 4.3.2.0 specification

IFC data underscores the importance of developing tools that streamline IFC data exploration and interpretation.

When building an AI parser for IFC data, it is essential to prioritize elements that capture both the spatial hierarchy and the physical fabric of a building. This includes spatial elements like *IfcBuilding*, *IfcSpace*, and *IfcBuildingStorey*, which define the organizational framework of the model. Moreover, architectural components such as *IfcWall*, *IfcSlab*, and *IfcDoor* are crucial as they represent the core physical elements of the building. As a proof of concept to validate the efficacy of Graph-RAG in parsing these relational aspects, we designed our experiments with two distinct settings:

1) **Holistic Parsing**: Parsing the entire IFC model with the entirety of its elements to test general retrieval capabilities across the full dataset.
2) **Targeted Parsing**: Focusing exclusively on a single essential element, specifically *IfcWall*, to evaluate how accuracy and retrieval precision are affected when the graph scope is narrowed to specific building components.

In our targeted parsing experiments with BIM-GRASP, we focus on parsing wall-related information from IFC files. This is because walls which are modelled using the *IfcWall* element in the IFC schema and its subtypes are particularly important as they serve structural, spatial, and thermal roles within a building.

Moreoever, our decision to specifically focus on the *IfcWall* element for Targeted Parsing was informed by consultations with our industrial partner, who provided valuable insights into the practical application of IFC models within the industry. According to the IFC schema shown in Figure 3, the *IfcWall* entity is a subtype of *IfcProduct* and is considered a fundamental building block, appearing in every standard schema subset and implementation level in the building model.[3] In practical workflows, walls define rooms and zones, influence HVAC and energy performance, support other building components (e.g., windows, roofs), and act as key carriers of fire ratings, acoustic properties, and insulation systems [12]. Furthermore, they are central to critical downstream tasks such as cost estimation, construction sequencing, and maintenance operations [26].

Accessing and interpreting these attributes through automated methods unlocks powerful insights for designers, engineers, and facility managers. Thus, prioritizing wall properties enables the development of an intelligent parser that can answer the complex queries necessary to understand how building elements interact structurally and spatially, as walls are inherently connected to numerous other elements in the building model. This highly interconnected nature of IFC elements, particularly *IfcWalls*, demonstrates that they are well-suited for representation as a knowledge graph for parsing purposes. In the following subsection, we provide a brief summary of the Graph-RAG technique used in our IFC parsing system.

[3]IFC 4.3.2.0 specification

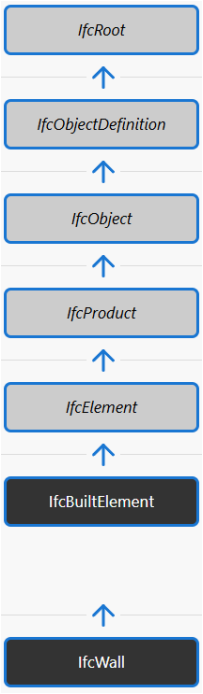


FIGURE 3. ***IfcWall*** **entity inheritance according to the IFC schema.**

B. GRAPH RETRIEVAL-AUGMENTED GENERATION

LLMs demonstrate remarkable proficiency in understanding natural language and generating coherent, contextually appropriate text. However, they are constrained by several limitations. Foremost among these is the absence of domain-specific knowledge, particularly in highly specialised fields. Moreover, because LLMs are trained on static datasets, they lack access to real-time updates and cannot incorporate proprietary or restricted information that falls outside their pre-training corpus. These deficiencies can result in hallucination, a phenomenon where the model produces information that is either inaccurate, incomplete, or entirely fabricated which ultimately reduces reliability in high-stakes or technical applications.

Retrieval-Augmented Generation (RAG) is an information retrieval method first introduced by Meta AI [27] to enhance the ability of LLMs to produce accurate answers from textual data. In its core formulation, both the input query and the text chunks from a target dataset are converted into vector representations using the same LLM (e.g. Llama 3 [28], GPT-4 [29], etc.). Representing all elements of the dataset as vectors equips the retrieval component with a richer contextual foundation, and hence enables it to locate the most relevant information with higher precision. In the RAG paradigm, this process begins with the retrieval module comparing the query vector to all vectors in the dataset to pinpoint the closest matches. These retrieved segments are then passed to the generative module, which uses a LLM to generate a coherent, natural-language response. Incorporating an external knowledge base into the workflow mitigates errors such as LLM hallucinations [30], [31], [32].

However, standard vector-based RAG architectures were essentially designed for unstructured text corpora, which introduces severe systemic limitations when applied to highly structured, schema-driven formats like IFC for several reasons [33]. First, vector RAG requires data to be divided into fixed-size sequence blocks (chunks) to generate embeddings. IFC models rely on deeply nested hierarchies (e.g., *IfcProject* → *IfcBuildingStorey* → *IfcWall* → *IfcRelDefinesByProperties* → *IfcMaterial*). When an IFC file is arbitrarily "chunked," the structural and spatial relationships could be disconnected. A chunk containing a wall's geometry may be entirely disconnected from the chunk detailing its parent building storey. Second, vector databases retrieve data based on semantic similarity rather than structural dependency [34]. If a user asks, "*What is the material of the 2nd-floor walls?*", a standard RAG system will retrieve chunks semantically related to "material," "2nd floor," and "wall." However, it cannot guarantee that the retrieved material actually belongs to the specific wall on that specific floor. This lack of deterministic linking causes hallucinations in complex multi-hop queries. To empirically validate these structural limitations, we conducted a preliminary vector-RAG baseline experiment on the Small House model using a standard chunking strategy. The full details of this baseline comparison are discussed in Section V.

Building on the RAG framework to overcome the aforementioned limitations of vector-based retrieval, Graph Retrieval-Augmented Generation (Graph-RAG) represents an advanced evolution that incorporates graph-structured knowledge sources to strengthen both information retrieval and multi-hop reasoning capabilities in LLMs [35]. Whereas conventional RAG systems primarily operate on unstructured text, Graph-RAG retrieves structured graph components (e.g., nodes, triples, paths, or entire subgraphs) that preserve explicit relationships between entities. This retrieval of interconnected graph segments provides richer contextual capabilities which resolves the reasoning gaps of standard RAG and improves the model's ability to retrieve context-aware outputs. Graph-RAG's ability to address complex, topology-driven information retrieval challenges has led to its adoption across a broad spectrum of applications. Beyond its use in downstream NLP tasks, it has been implemented in diverse domains such as e-commerce [36], legal analytics [37], and biomedicine [38]. However, its potential to navigate the deeply nested, object-oriented hierarchies of Building Information Models (BIMs) has remained largely unexplored.

In our study, Graph-RAG was applied to a pre-constructed knowledge graph derived from IFC data; the detailed process of constructing this graph from the raw IFC model will be thoroughly explained in the methodology section. To support this retrieval process, our framework employs the Neo4j graph database and its declarative query language, Cypher [39]. The Cypher queries enable precise specification of subgraphs of interest within the IFC-derived knowledge graph and facilitates seamless integration between structured data and generative reasoning. A short summary of the Cypher language is mentioned in the next section.

C. CYPHER QUERY LANGUAGE

Cypher is a declarative query language developed for property graphs, originally as part of the Neo4j database, and is now widely adopted across graph data management systems [40]. Cypher is pattern-oriented and enables users to express complex graph structures through intuitive "ASCII art" syntax (e.g., $(a) - [r] \rightarrow (b)$), to capture nodes, relationships, and their properties. Queries are composed of clauses that operate linearly to traverse and extract relevant subgraphs. This makes Cypher well suited for retrieving interconnected entities and their relationships, such as walls, spaces, or constraints within IFC-based knowledge graphs. For example, to find the materials of walls located on a specific floor, the graph traversal can be expressed as:

```
MATCH (s:BuildingStorey {name: '2nd
    Floor'})-[:CONTAINS]->(w:Wall)-[:
    HAS_MATERIAL]->(m:Material) RETURN m.
    name
```

In the context of our work, Cypher serves as a mechanism for subgraph retrieval. It complements the Graph-RAG by enabling the extraction of structured relational patterns that can later be translated into natural language responses. A detailed explanation of how this pipeline works is provided in the next section.

IV. METHODOLOGY

The IFC parsing method involves two stages: 1) transforming the IFC model into a graph, and 2) using an LLM to interpret the graph and answer user queries. Theses two stages are explained below.

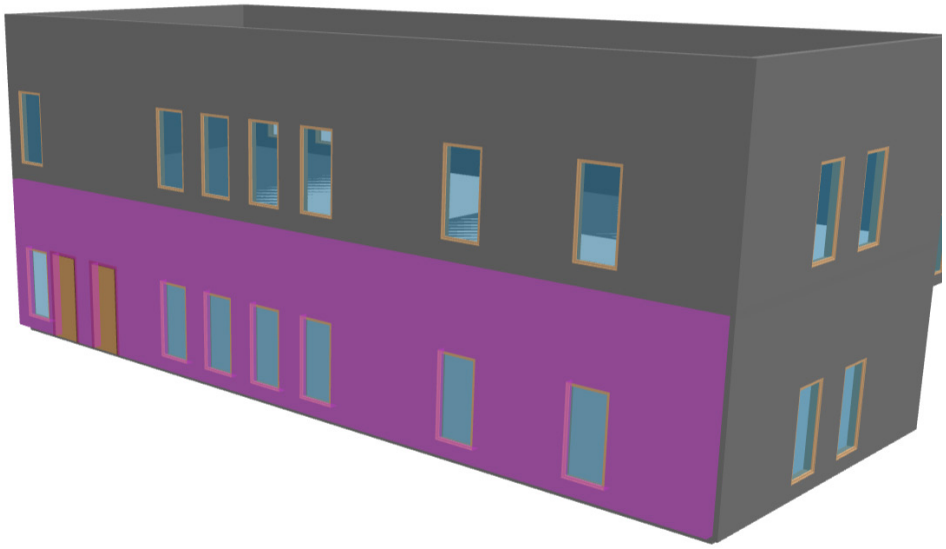
A. STAGE 1: GRAPH DATABASE GENERATION

To systematically represent the IFC data and evaluate how graph complexity impacts retrieval accuracy, we established a parsing protocol using the IfcOpenShell Python library [41]. This open-source toolkit enables the efficient traversal of the IFC schema, allowing us to extract entities, attributes, and relationships by iterating through entity IDs in ascending order. For this study, we generated two distinct graph topologies to test the system under different structural constraints:

1) **Holistic Graph (HG):** This configuration represents the entire IFC model as a single global graph, G, which serves as the foundational structure for the BIM-GRASP framework. Formally, the graph is defined as $G = (V, E)$, where V represents the set of vertices (or nodes) and E represents the set of edges. Each vertex $n \in V$ corresponds to a unique IFC entity extracted from the model, such that $V = \{n_1, n_2, \ldots, n_N\}$. An edge, $e \in E$, is established between two nodes, n_i and n_j, if entity n_i contains an explicit reference to

```
#2736= IFCWALLSTANDARDCASE('2cXV28XOjE6f6irgi0COkg',#42,'Basic Wall:Outside wall:352966',$,'Basic Wall:Outside wall',#2689,#2734,
#2739= IFCQUANTITYLENGTH('Height',$,$,3850.);
#2740= IFCQUANTITYLENGTH('Length',$,$,20000.);
#2741= IFCQUANTITYLENGTH('Width',$,$,200.);
#2742= IFCQUANTITYAREA('GrossFootprintArea',$,$,3.99999999999996);
#2743= IFCQUANTITYVOLUME('NetVolume',$,$,15400000000.);
```

(a) IFC snippet



(b) Wall representation in building model

FIGURE 4. **(a) A sample data section of an IFC file, and (b) the corresponding physical element highlighted within the building model.**

entity n_j within its attribute set. The formal relationship is expressed as:

$$\begin{aligned} G &= (V, E), \\ V &= \{n_1, n_2, \ldots, n_N\}, \\ E &= \{(n_i, n_j) \mid ID(n_j) \in \text{Att}(n_i)\}. \end{aligned} \tag{1}$$

where $\text{Att}(n_i)$ denotes the set of attributes associated with entity n_i, and $ID(n_j)$ represents the unique IFC identifier of entity n_j. This approach produces a dense, highly interconnected graph that preserves all semantic and geometric relationships within the model, enabling the Graph-RAG component to navigate the complete building topology.

2) **Targeted-Element Graph (TG):** The second configuration narrows the scope exclusively to *IfcWall* elements, applying the same mathematical formulation as the HG (Equation 1). However, the node set is restricted entirely to wall entities. Each node represents an individual wall and is populated with its specific attributes (e.g., *IsExternal*, *FireRating*, *ThermalTransmittance*). Edges are established only between pairs of wall nodes that share a direct topological relationship (e.g., connected via *IfcRelConnectsPathElements*)).[4]

For both configurations, graph construction follows a standardized mapping from the IFC schema to ensure compatibility with our retrieval pipeline. Figure 4 illustrates an example of how we apply our formal mathematical framework on a snippet of an IFC file shown in Figure 4(a) which corresponds to the specific wall highlighted in purple within the building model in Fig. 4(b). Following the vertex set definition $V = \{n_1, n_2, \ldots, n_N\}$, each entity preceded by a hash symbol (underlined in Figure 4(a)) is instantiated as a unique node $n \in V$. The set includes $\{n_{42}, n_{2736}, n_{2689}, n_{734}\}$. The primary entity n_{2736} is created with the label `IfcWallStandardCase`, while n_{2739} through n_{2743} are instantiated as attributes to this node representing specific quantity types, `IfcQuantityLength`. The attribute set $\text{Att}(n_i)$ for this node stores metadata and literal values, such as the wall's height (3850.0) and length (20000.0), etc. According to the rule $E = \{(n_i, n_j) \mid ID(n_j) \in \text{Att}(n_i)\}$, edges are established by identifying explicit references within these attribute sets. For the wall entity n_{2736}, directed edges are created to connect this wall to other adjacent walls or building elements to which it is related. This includes structural and geometric dependencies, such as the connections to nodes n_{2689} and n_{2734} which define the wall's spatial placement and physical representation in the building model.

Moreover, we define a property classification scheme that groups attributes into four core categories: **Geometry**, **Constraints**, **Connections**, and **Material**. These four categories represent the core descriptive dimensions of any physical building element and provide sufficient information to support the widest range of downstream applications [42]. Geometry properties (e.g. length, height, width, footprint, volume) are essential for accurately reconstructing the wall's shape in spatial analysis or visualisation tools which

[4] IfcWall Documentation

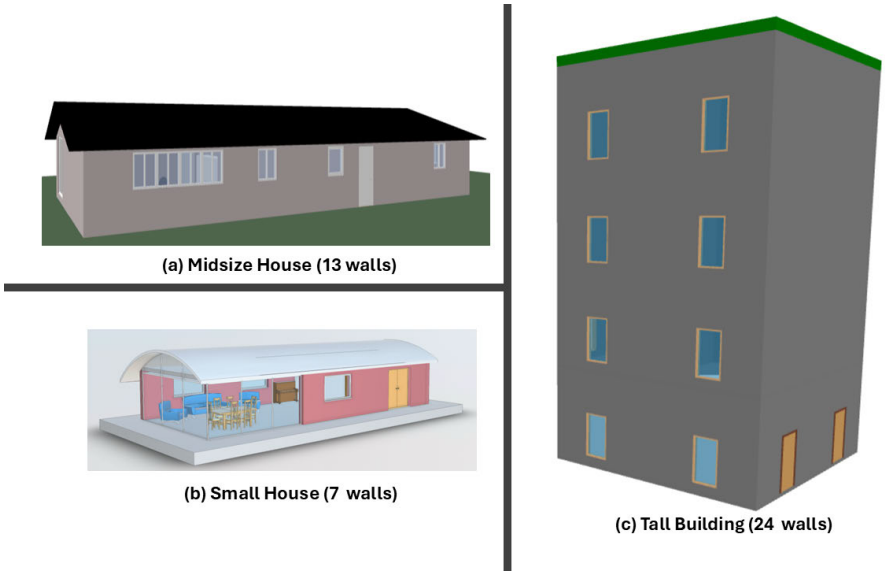


FIGURE 5. IFC viewer of the three building models used in the experiments.

enables precise drawing, measurement, and clash detection. Constraint properties (e.g. offsets, base/top constraints, room bounding) preserve spatial relationships and design intent by defining how the wall is positioned relative to other elements and building levels. Connection properties (e.g. connected elements, IDs, connection type) capture relational integrity which describes how the wall interacts with surrounding structural and non-structural components for dependency tracking, impact assessment, and graph-based navigation. Material properties (e.g. structural material, finishes, U-value, thermal resistance) define the wall's physical performance in energy modelling, structural calculations, and sustainability assessments which can allow simulations of thermal, acoustic, and environmental behaviour of the building elements. Other IFC property sets, such as revision history, authoring software, or project metadata, while useful for documentation and project management, do not directly influence the physical, spatial, or performance definition of the wall. Consequently, we focus on these four categories as the minimum necessary data to meaningfully represent a wall for the underpinning critical processes from BIM coordination to compliance verification. Appendix shows mapping between raw IFC data for these four categories and their semantic meaning in the context of wall elements.

For our experiments, we evaluated our system across four distinct building models of varying structural complexity and relationship scale. This approach allowed us to stress-test the LLM's capability to retrieve information across a spectrum of simple to highly intricate knowledge graphs. These models include open source IFC data of a single-family dwelling with 7 walls (Small house), a suburban residence with 13 walls (Midsize house), a multi-storey building with 24 walls (Tall building), and a proprietary commercial BIM project with 365 walls (Industry Case Study). We deliberately combined open-source BIM data with an authentic dataset provided by our industry partner to establish a dual-phase validation process. The open-source data for the Small house, Midsize house, and Tall building serves as a standardized baseline benchmark. Because these models are publicly accessible, they provide clean, well-documented schemas that allow for reproducible controlled testing of our LLM's basic retrieval mechanisms.

However, synthetic or idealized open-source models often lack the structural irregularities and modeling noise found in daily engineering practice. To bridge the gap between academic theory and practical deployment, we experimented with the Industry Case Study building which is sourced directly from an active commercial project by our industry partner. Testing the system against this real-world asset ensures that our graph-based retrieval approach can handle the messy, large-scale data environments encountered by industry professionals. Figure 5 shows a 3-dimensional view of the open-source building models visualized by the Sortdesk IFC viewer[5]; due to page space limitations, the 345-wall Industry Case Study building is omitted from this visualization.

Data extraction begins by parsing IFC files with the Python library `ifcopenshell`, which efficiently handles the hierarchical, schema-rich structure of IFC models [41]. For targeted parsing, we focus on the *IfcWall* entity and its subtypes—such as *IfcWallStandardCase*, which adheres to stricter geometric rules, and *IfcCurtainWall*, a specialized façade wall. We also capture the bidirectional relationships between walls, represented through IFC's *IfcRelConnectsElements* schema. Each wall is identified both as a relating and a related element within these connections.

[5]https://viewer.sortdesk.com/

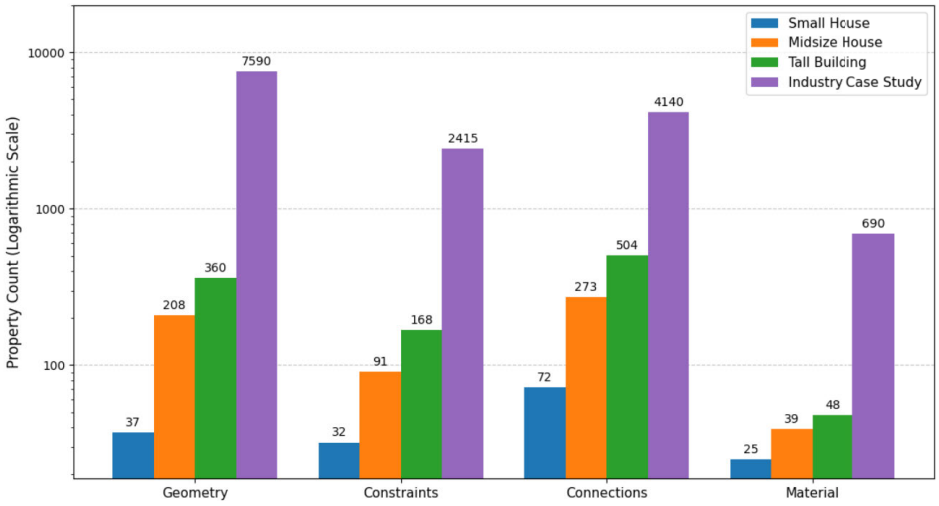


FIGURE 6. ***IfcWall*** **property distribution by category for each building model.**

Properties and quantities are gathered from property sets (psets) like *Pset_WallCommon* and *BaseQuantities*, focusing on the four key categories: Geometry, Constraints, Connections, and Material. The extracted data is then transformed into a flattened intermediate representation that preserves essential relationships and metadata for each wall element within these categories. Figure 6 illustrates the distribution of these extracted IFC properties for each of the evaluated building models. A logarithmic scale is employed on the vertical axis to accurately represent the disparity in data volume between the baseline open-source models (Small house, Midsize house, and Tall building) and the proprietary Industry Case Study. As shown in the Figure, the structural complexity of the real-world commercial asset results in a large expansion of the knowledge graph, particularly within the Geometry and Connections categories, which peak at 7,590 and 4,140 properties respectively. This scaling supports our approach to constructing knowledge graphs with progressively greater complexity to provide a robust testbed to assess the LLM's retrieval capabilities across a spectrum of building model intricacies.

B. STAGE 2: GRAPH-RAG WITH LARGE LANGUAGE MODEL

In this step, we detail the core Graph RAG Component of our framework. The workflow begins when an Input Query is received. To process this, we implement a dual-tool retriever agent driven by a pre-trained large language model (LLM). The first tool handles Knowledge Enhancement, where the LLM uses its foundational, pre-trained understanding of standard IFC schemas to contextualize the user's intent and generate baseline architectural context. The second tool performs Query Enhancement which translates the natural language input into executable Cypher queries tailored to the specific topological structure of the IFC Graph Database. The agent then executes a Graph Retrieval by querying the database, extracting relevant sub-graphs and element properties, and passes these specific Retriever Results back to the LLM.

To ensure multi-turn interactions, we implement a callback function that maintains conversational memory, allowing the system to handle follow-up Input Queries coherently. Also, strict prompt engineering bounds the Graph RAG Component as the agent is explicitly instructed to confine its Knowledge Enhancement strictly to the IFC scheme. This prevents out-of-scope answers and ensuring domain relevance.

We evaluated several LLM configurations for this component, including open-source models (deepseek-coder-6.7b-instruct [43]), zero-shot baselines (GPT-4 and GPT-4.1), and an optimized version of GPT-4.1 which utilizes a dynamic In-Context Learning (ICL) library. The ICL optimises our Query Enhancement stage beyond zero-shot capabilities without changing the model's underlying weights. We achieved ICL by framing Cypher generation as a semantic translation problem. Instead of performing parameter-updating fine-tuning on a massive dataset, we curated a JSON-based few-shot context library. This library consists of 20 parallel examples across different IFC element types and properties. Each example pairs an IFC term with its natural language description, a representative Input Query, the corresponding Cypher template, the expected Retriever Results, and the ideal Final Answer. Table 1 showcases representative examples from this prompt library. Due to space limitations, the complete set of the ICL cannot be included in this manuscript. However, the complete JSON-based ICL prompt library is publicly available in our GitHub repository.

Guided by this ICL technique, the Graph RAG Component accurately the full pipeline: it formats the Cypher code, triggers the Graph Retrieval, and transfers the raw database output to a Natural Language Translation. This final translation stage converts the complex relational data into an intuitive, user-friendly Final Answer. By utilizing

TABLE 1. Samples of cypher-query prompt context library.

Question	How many walls exist in this building?
Cypher	MATCH (n:IfcWall) RETURN COUNT(n) AS WallCount
Context	{'WallCount': 7 }
Response	There are 7 walls in the building.
Question	What are the types of these walls?
Cypher	MATCH (w:Element) WHERE w.type IN ['IfcWall', 'IfcWallStandardCase','IfcCurtainWall'] RETURN w.id AS Wall_ID, w.name AS Wall_Name, w.type AS Wall_Type
Context	{'Wall type ': IfcWall.type('types: ['IfcWall', 'IfcWallStandardCase', 'IfcCurtainWall']') }
Response	The wall types are IfcWallStandardCase.
Question	"Which wall has the largest volume on the top floor?"
Cypher	MATCH (w:Wall) WHERE w.'Top Constraint' = '(?i).*roof.*' OR w.'Top Constraint' = '(?i).*top.*' AND w.'Dimensions Volume' IS NOT NULL RETURN w.name AS WallName, w.'Top Constraint' AS TopLevel, toFloat(w.'Dimensions Volume') AS Volume ORDER BY Volume DESC LIMIT 1
Context	{'w.Top Constraint = '(?i).roof.' OR w.Top Constraint = '(?i).top., w.Dimensions Volume IS NOT NULL' }
Response	The wall name is "Basic Wall: Exterior - Brick 285330" on "Roof Level" with volume: "145.67"

structured prompts rather than weight-updating fine-tuning, this approach eliminates the risks of "catastrophic forgetting" of general knowledge and "data leakage" (overfitting). Also, it improves the model's ability to generalize correct patterns and minimises hallucinations to enhance response reliability. To standardize Cypher query generation and ensure consistent Retriever Results, we strictly controlled the model's inference hyperparameters. Specifically, the temperature was set to 1.0 to regulate randomness (for purely deterministic structural parsing, this value could be minimized). We configured nucleus sampling (*top_p*) to 1.0 to allow the model to consider the top 100% of the token probability mass during token selection. Also, the Reasoning Effort parameter was set to 'Medium' to balance processing speed with the depth of logic required for accurate graph traversal.

C. MULTI-HOP REASONING WITH DIFFERENT LANGUAGE MODELS

We breakdown the assessment of the BIM-GRASP performance not only in terms of the four IFC categories mentioned above but also in terms of query complexity. Thus, we construct a dataset of 100 queries divided equally into: Simple (single-hop reasoning), Moderate (two-hop reasoning), and Complex (multi-hop reasoning). A Simple question asks a straightforward element such as "What is the height of this wall?" which only requires accessing the *BaseQuantities.Height* property of a single wall. A multi-hop question requires connecting multiple nodes or relationships in the knowledge graph to answer correctly. An example of a multi-hop would be as follows:

→ Question: "*Which walls have a different material than the walls they connect to?*"
- Hop 1: Extract all connected walls in the IFC model using *IfcRelConnectsElements*
- Hop 2: Find material of each wall in a node community by *IfcMaterial*, *Finishes.Structural Material*, or *IfcMaterialLayerSet*
- Hop 3: Compare output of Hop 2 to extract dissimilar elements
- Hop 4: Return wall IDs that have different material

Multi-hop questions test whether a system understands logical dependencies between elements. Also, it assesses the ability of the agent to traverse and reason over the structure of the building and to combine filters, rankings, and relationships between elements. Parsing this information is important as it is commonly needed in real-world use cases such as energy analysis, safety inspection, and spatial planning in AECO.

To evaluate the performance of each LLM with our BIM-GRASP retriever agent, we computed precision (P), recall (R), and the F1 score for each property category and building. To derive these metrics, we classify the retriever's outputs as follows:

- True Positive (TP) refers to cases when the correct value is returned (e.g., the query is "What is the thermal resistance of this wall?" and the correct value is provided).
- True Negative (TN) refers to cases when the information is absent from the IFC model and the system appropriately responds with answers such as "I don't know".
- False Positive (FP) refers to cases when an incorrect value is returned (e.g., the wrong footprint area).
- False Negative (FN) refers to cases when the retriever agent either does not give any answer or responds with answers such as 'I don't know' despite the information being available in the knowledge graph.

Also, to better understand the reasoning capabilities of each model with the retriever agent, we estimated accuracy based on query difficulty, as it provides a meaningful gradient for evaluating how well language models handle increasing semantic and structural complexity. We define the average accuracy as follows:

$$A_{\text{avg}} = \frac{1}{3} \sum_{d \in \{s,m,c\}} \frac{TP_d + TN_d}{TP_d + TN_d + FP_d + FN_d} \tag{2}$$

where the average accuracy A_{avg} of an LLM across varying question difficulty levels is computed by aggregating its performance on Simple (s), Moderate (m), and Complex (c) questions. For each difficulty level $d \in \{s, m, c\}$, accuracy is defined using standard classification metrics where TP_d, TN_d, FP_d, and FN_d represent the true positives,

Algorithm 1 Algorithm: BIM-GRASP – A Graph-RAG Pipeline for Semantic Parsing of IFC Building Models

Input: IFC model file $\mathcal{F}$
Output: Natural language answer $\mathcal{A}$ to user query $\mathcal{Q}$
procedure BIM-GRASP($\mathcal{F}$, $\mathcal{Q}$)
  Step 1: Parse IFC Model
  Extract a set of elements $E = \{e_1, e_2, \ldots, e_n\}$ from $\mathcal{F}$
  For each element e_i, extract attributes $\mathcal{P}_i = \{p_1, p_2, \ldots, p_k\}$
  Step 2: Construct Knowledge Graph
  Build a graph $\mathcal{G} = (V, R)$
  where $V = E$ are nodes and $R = \{r_1, r_2, \ldots\}$ are inter-element relations
  Step 3: Create Few-Shot Context Library
  Construct dataset $\mathcal{D} = \{(\mathcal{Q}_i, c_i)\}$, where $\mathcal{Q}_i$ is an NL question and c_i is a Cypher query
  Step 4: Construct Prompt Template
  Inject context library $\mathcal{D}$ into the LLM prompt to guide the mapping $\mathcal{Q}_i \mapsto c_i$
  Step 5: Generate Cypher Query
  Given input query $\mathcal{Q}$, predict Cypher query $c = \mathcal{M}(\mathcal{Q})$
  Step 6: Retrieve Information
  Execute c on $\mathcal{G}$ to retrieve structured results r
  Step 7: Answer Generation
  Generate answer $\mathcal{A} = \mathcal{M}(r)$ using the LLM's NLG capabilities
  return $\mathcal{A}$
end procedure

true negatives, false positives, and false negatives, respectively. The overall average accuracy is then calculated as the mean of these three values. This formulation provides a balanced evaluation of the model's predictive performance across question types. This ensures that accuracy is not biased toward any single difficulty level.

To support reproducibility, we include an Algorithm 1 detailing the steps required to implement our BIM-GRASP framework so as to enable the research community to adapt our approach for other IFC elements according to specific use cases. The underlying algorithm of BIM-GRASP offer a generic, extensible structure that can be applied to develop similar parsing systems for any BIM element as needed.

V. EXPERIMENTAL RESULTS AND DISCUSSION

Before evaluating the Graph-RAG framework, we established a standard vector-RAG baseline using the Small House model (7 walls). The IFC data was processed with a chunk size of 1500 characters and an overlap of 200 characters to minimize data loss across boundaries and used GPT-4.1 as the retrieval LLM. The vector-RAG system yielded an accuracy of just 22.73% and an exceptionally high False Positive Rate of 96.43%. While the baseline could occasionally retrieve correct answers for one-hop queries regarding isolated properties (e.g., element volume or dimensions), it consistently hallucinated when tasked with relational or comparative queries (e.g., structural connections). Because returning fabricated data is very dangerous in a BIM context, these baseline results empirically demonstrated that vector-RAG cannot guarantee the topological correctness required for IFC parsing. Consequently, we focused our primary evaluations exclusively on the Graph-RAG approach.

Table 2 presents results of the LLM experiments across the three building models with our Graph-RAG approach. The metrics detail precision, recall, and F1-scores across four building models and four semantic categories. The baseline zero-shot GPT-4 model exhibits poor recall across most test cases. This indicates failure to isolate complete data paths during the graph retrieval stage. Deepseek performs competitively on Small House queries (F1 ≈ 0.84 across most categories), but its performance drops significantly on Tall Building connections (F1 $= 0.26$), indicating challenges in multi-hop reasoning and relational depth.

Upgrading the core reasoning engine to GPT-4.1 yields substantial improvements even without contextual examples. The zero-shot GPT-4.1 model demonstrates higher precision and recall across the baseline datasets. It effectively parses intermediate geometric and constraint queries. However, it still struggles with highly interconnected topological data. This limitation is evident in the connections category for the tall building (F1 score of 0.49). These metrics highlight the limitations of relying only on pretrained knowledge for complex spatial parsing.

The query enhancement stage achieves best performance when augmented with the In-Context Learning (ICL) library. The GPT-4.1 (ICL) configuration consistently outperforms all other evaluated approaches. It records the highest F1 scores in nearly every category and building scale. By providing structured examples, the system effectively stabilizes the graph retrieval process. This methodology eliminates the recall degradation observed in the baseline zero-shot models.

TABLE 2. Evaluation of LLMs on IFC question-answering by category and building type. Bold indicates the best performance.

Model	Building	Geometric			Constraints			Connections			Material		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
GPT-4	Small House	0.71	0.17	0.28	0.77	0.25	0.38	0.44	0.25	0.32	1.00	0.20	0.34
GPT-4	Midsize House	0.89	0.86	0.87	0.84	0.82	0.83	0.88	0.89	0.89	0.80	0.78	0.79
GPT-4	Tall Building	1.00	0.40	0.57	1.00	0.25	0.40	1.00	0.50	0.66	0.66	0.25	0.36
GPT-4	Industry Case Study	0.60	0.10	0.18	1.00	0.18	0.31	1.00	0.38	0.55	1.00	0.14	0.25
GPT-4.1	Small House	0.95	0.70	0.80	1.00	0.82	**0.90**	0.88	0.75	0.81	0.93	0.65	0.76
GPT-4.1	Midsize House	0.78	0.81	0.80	0.81	0.84	0.82	0.89	0.80	0.84	0.87	0.73	0.80
GPT-4.1	Tall Building	1.00	0.60	0.75	1.00	0.77	0.87	0.50	0.40	0.49	1.00	0.60	0.75
GPT-4.1	Industry Case Study	0.62	0.89	0.73	0.82	0.90	0.86	0.60	1.00	0.75	0.80	1.00	0.89
GPT-4.1(**ICL**)	Small House	1.00	0.93	**0.96**	0.89	0.80	0.84	1.00	0.89	**0.94**	1.00	0.89	**0.94**
GPT-4.1(**ICL**)	Midsize House	1.00	0.91	**0.95**	1.00	1.00	**1.00**	1.00	0.86	**0.92**	1.00	0.72	**0.84**
GPT-4.1(**ICL**)	Tall Building	1.00	1.00	**1.00**	1.00	0.85	**0.92**	**1.00**	0.75	**0.86**	1.00	0.80	**0.89**
GPT-4.1(**ICL**)	Industry Case Study	0.89	0.80	**0.84**	0.81	1.00	**0.90**	1.00	0.70	**0.82**	1.00	1.00	**1.00**
Deepseek	Small House	0.75	0.81	0.78	1.00	0.72	0.84	0.88	0.80	0.84	0.88	0.80	0.84
Deepseek	Midsize House	0.70	0.63	0.66	1.00	0.75	0.85	0.50	0.30	0.37	1.00	0.63	0.77
Deepseek	Tall Building	0.77	0.58	0.66	0.88	0.72	0.80	0.33	0.22	0.26	0.66	0.75	0.70
Deepseek	Industry Case Study	0.47	0.47	0.47	1.00	0.44	0.62	1.00	0.50	0.67	0.75	0.75	0.75

It is important to note that the ICL methodology scales well to the production-grade industry case study. It achieves perfect F1 scores of 1.00 for material properties in the largest graph environment. It also maintains high accuracy for complex structural constraints. These evaluation metrics confirm that the structured prompt library successfully guides the system architecture. It also indicates that the BIM-GRASP system with our best model is capable of parsing large scale IFC real-world data and enables the natural language translation stage to deliver reliable final answers regardless of underlying graph complexity.

Additionally, Figure 7 illustrates the average accuracy of the four language models GPT-4, GPT-4.1, GPT-4.1(**ICL**), and Deepseek across three levels of question difficulty: Easy, Moderate, and Hard. As outlined in the methodology, query difficulty is determined by the number of reasoning hops required by the agent retriever to navigate the knowledge graph and retrieve the correct answer. As shown in the figure, GPT-4.1(**ICL**) demonstrates the strongest overall performance, achieving perfect accuracy on easy queries and maintaining relatively high scores on moderate and hard ones (83% and 77% respectively). This reflects its robustness in handling complex IFC-related reasoning. GPT-4.1 performs well on easy and moderate queries but drops sharply to 55% on hard ones, while GPT-4 shows consistently lower scores across all levels. Deepseek exhibits the steepest decline, falling from 74% on easy queries to just 30% on hard ones which indicates limited generalization to more challenging semantic and relational tasks. Notably, only GPT-4.1(**ICL**) remains above the 70% threshold across all difficulty levels, whereas Deepseek fails to meet the 50% mark on moderate and hard queries. These empirical thresholds offer a practical benchmark for evaluating deployment readiness, especially in high-stakes BIM environments where precision and consistency are critical. The results indicate that in context learning on our parallel IFC terms, Cypher queries, and the natural semantic interpretation of domain-specific terminology proved to be an effective strategy particularly in boosting accuracy on multi-hop reasoning tasks that require layered inference and contextual alignment across different reasoning levels.

A. ERROR ANALYSIS

We conducted a qualitative error analysis of the LLM outputs to better understand the cases in which the models failed to provide correct answers to user queries and cases where it performed exceptionally well on multi-hop questions. Table 3 presents representative examples of LLM failure modes alongside instances where the model successfully resolved user queries. Our observations are summarised as follows:

1) FALSE POSITIVES AND PRECISION

A notable finding is that all LLMs rarely produced false positives, that is, they seldom returned an incorrect value as an answer. This explains why precision scores are generally higher than recall across all models.

2) MULTI-HOP REASONING AND FALSE NEGATIVES

The zero-shot versions of GPT as well as Deepseek struggled with multi-hop reasoning, often resulting in false negatives by responding with "I don't know" or "The information is not available". We consider these answers as false negatives in cases where the information is in the graph but the system is not able to retrieve it. Interestingly, these same queries could often be resolved if the reasoning process was broken down into simpler steps. For example, all baseline zero-shot models frequently failed when asked directly:

> "Which wall connected to this wall has the highest volume?"

Yet, when the query was decomposed into incremental steps—first asking which walls are connected, then their volume, and finally which has the highest value—the model was able to produce the correct answer.

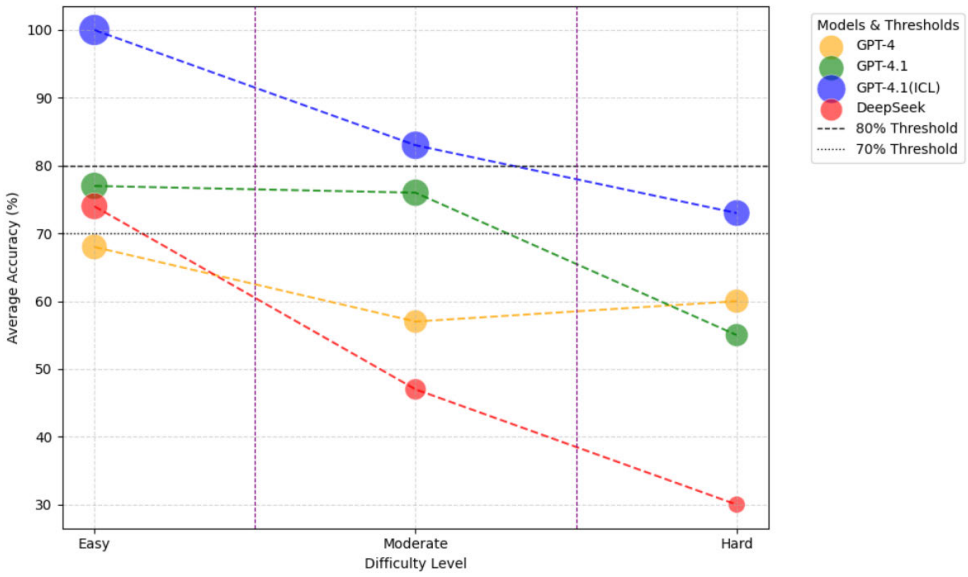


FIGURE 7. Model average accuracy across question difficulty levels.

3) GPT-4.1(**ICL**) AND GLOBAL REASONING

The GPT-4.1 (**ICL**) model demonstrated stronger capabilities in handling multi-hop reasoning that required parsing the entire graph. These are tasks that are particularly challenging for conventional IFC parsing tools. For example, the question:

> "Which exterior wall in the building has the highest Dimensions Volume?"

the The GPT-4.1 (**ICL**) model was able to correctly infer that the query required global graph traversal and gave the correct answer.

4) COMPARATIVE AND RELATIONAL REASONING

More complex reasoning tasks were successfully handled mostly by GPT-4.1 (**ICL**). Fore example, it correctly answered comparative and relational queries such as:

> "Which wall has a different material than the walls connected to it?"

and

> "Find the name of the wall connected to Basic 285459 that has the lowest thermal resistance."

These examples demonstrate the model's ability not only to retrieve information but also to perform comparisons and relative evaluations. These are capabilities that extend well beyond the functions of traditional IFC parsers. The zero-shot baseline models, on the other hand, struggled to give the correct answer to such relational reasoning queries.

5) RECOVERY FROM REASONING BLOCKS

Zero-shot LLMs often reached reasoning blocks and failed to recover specially when queries shifted from easy to complex and then back to easy. In such scenarios, these models frequently became "stuck" and continued to respond with "I don't know," even for simple queries they had previously answered correctly (e.g., "What is the type of this wall?"). This behavior highlights the advantage of ICL, which provides greater robustness and adaptability in mixed-sequence reasoning tasks.

6) STRUCTURED OUTPUTS AND MATHEMATICAL COMPUTATION

A key advantage of the BIM-GRASP framework, when paired with LLMs, is its ability to present retrieved information in a structured format, for example, as tabular data pairing wall names with property values. We found that both GPT-4.1 and GPT-4.1(**ICL**) was able to perform on-the-fly numerical computations, such as responding to:

> "Give the name of each wall beside its Dimensions Volume and approximate the volume to two decimal points."

This feature is particularly valuable, since many IFC properties contain long decimal values. Furthermore, the ability to incorporate more complex mathematical operations into queries makes BIM-GRASP a far more powerful tool than traditional IFC parsing approaches.

Finally, it is worth noting that across all models, material-related queries tend to yield higher F1 scores, likely due to the more direct attribute mappings in IFC schemas. In contrast, connection-based queries show greater variability, reflecting the complexity of spatial and relational reasoning.

B. END-TO-END CASE STUDIES

To provide full transparency into the inner workings of our Graph-RAG pipeline and address the mechanics of our retrieval framework, we detail two end-to-end case studies: a successful multi-hop retrieval and a failure case.

The first case is Q4 in Table 3. The user asks, "*Which wall connected to this wall has the longest Dimensions Length?*" This query, alongside structured building data

TABLE 3. Examples of correct (green) and incorrect (red) LLM responses across different question types.

#	User Question	Correct Value	Cypher Query	LLM Response
Q1.	What is the Location Line of this wall?	Wall Centerline	MATCH (w:Wall {name: 'Basic Wall:Innervägg Paroc:1298036'}) RETURN w.`Location Line` AS locationLine	The Location Line of "Basic Wall:Innervägg Paroc:1298036" is "Wall Centerline".
Q2.	Are all walls connected to wall Basic Wall-Innervägg Paroc:1298038 of the same type?	Yes	MATCH (w:Wall {name: "Basic Wall:Innervägg Paroc:1298038"})-[:CONNECTED]-(other:Wall) WITH COLLECT(DISTINCT other.type) AS types	Yes, all walls connected to "Basic Wall:Innervägg Paroc:1298038" are of the same type, specifically IfcWall-StandardCase.
Q3.	How many connections does wall Basic Wall:Innervägg Paroc:1298038 have?	2	MATCH (w:Wall {name: 'Basic Wall:Innervägg Paroc:1298038'})-[r:CONNECTED]-(other:Wall) RETURN COUNT(r) as num	The wall "Basic Wall:Innervägg Paroc:1298038" has 2 connections.
Q4.	Which wall connected to this wall has the longest Dimensions Length?	Basic Wall:Yttervägg Paroc:1298028	MATCH (w:Wall) RETURN w ORDER BY toFloat(w.`Dimensions Length`) DESC	The wall named "Basic Wall:Yttervägg Paroc:1298028" has the longest Dimensions Length, which is 22060.0.
Q5.	Which wall has the smallest Gross Footprint Area?	Basic Wall:Innervägg Paroc:1298037	MATCH (w:Wall) RETURN w.name AS WallName, w.`Gross Footprint Area` AS GrossFootprintArea ORDER BY w.`Gross Footprint Area` ASC LIMIT 1	The wall with the smallest Gross Footprint Area is "Basic Wall:Innervägg Paroc:1298037" with a Gross Footprint Area of 0.3825.
Q6.	What type of relationship connects wall Basic Wall:Innervägg Paroc:1298036 to Basic Wall:Innervägg Paroc:1298592?	No connection	MATCH (w1:Wall {name: 'Basic Wall:Innervägg Paroc:1298036'})-[rel:CONNECTED]-(w2:Wall name: 'Basic Wall:Innervägg Paroc:1298592') RETURN type(rel) AS RelationshipType, rel	I do not have information about the type of relationship connecting "Basic Wall:Innervägg Paroc:1298036" to "Basic Wall:Innervägg Paroc:1298592".
Q7.	Is wall Basic Wall:Yttervägg Paroc:1298030 connected to more than 2 walls?	Yes- 3 walls	MATCH (w:Wall name:'Basic Wall:Yttervägg Paroc:1298030')-[:CONNECTED]-(other:Wall) RETURN count(other) as connectedwalls	Yes, the wall "Basic Wall:Yttervägg Paroc:1298030" is connected to more than 2 walls; it is connected to 3 walls.
Q8.	What are the type of connections to this wall?	IfcRelConnects PathElements	MATCH (w:Wall {id: '2DedXznHnDaeAWsrTBqBj'})-[r:CONNECTED]-(connectedWall:Wall) RETURN type(r) AS ConnectionType, r, connectedWall	I'm sorry, but I do not have information about the types of connections to this wall based on the provided context.
Q9.	What is the total Dimensions Area of walls connected to wall Basic Wall:Innervägg Paroc:1298592?	14.267	MATCH (w:Wall name: 'Basic Wall:Innervägg Paroc:1298592')-[:CONNECTED]-(adj:Wall) RETURN sum(adj.`Dimensions Area`) as TotalConnectedDimensionsArea	Error: code: Neo.ClientError.Statement.TypeError message: sum() can only handle numerical values, duration, and null. Got String("9.04799999999999")
Q10.	what is the type of this wall?	IfcWall StandardCase	MATCH (w:Wall) WHERE w.id = '2DedXznHnDaeAWsrTBqBj' RETURN w	The type of the wall with ID 2DedXznHnDaeAWsrTBqBj is "Basic Wall:Innervägg Paroc:1298034".

supplied by the IFC Graph Database, is fed into the core Graph RAG Component. Within this module, the system drives two parallel optimizations. First, Query Enhancement utilizes relational data retrieved from the graph to refine the user's search intent regarding connected elements and quantitative properties. Simultaneously, Knowledge Enhancement deploys the Generative LLM's foundational pre-trained knowledge to contextualize the query with broader domain awareness regarding building dimensions. Following these enhancements, the system executes a Graph Retrieval by formulating the Cypher query: `MATCH (w:Wall) RETURN w ORDER BY toFloat(w.`Dimensions Length`) DESC`. This step isolates the pertinent data paths in the graph, yielding specific Retriever Results that identify the wall "Basic Wall:Yttervägg Paroc:1298028" with a length value of 22060.0. In the final stage, the framework performs a Natural Language Translation on this retrieved graph data, converting the relational insights into the user-friendly Final Answer: "*The wall named 'Basic Wall:Yttervägg Paroc:1298028' has the longest Dimensions Length, which is 22060.0.*"

The second case illustrates a representative failure show in Q8 in Table 3. The user asks, "*What are the type of connections to this wall?*" Similar to the successful case, this query is fed into the Graph RAG Component. During Query Enhancement and Knowledge Enhancement, the system attempts to contextualize the request for specific connection topologies, which in a standard IFC schema should target entities like `IfcRelConnectsPathElements`. However, during the subsequent Graph Retrieval stage, the system

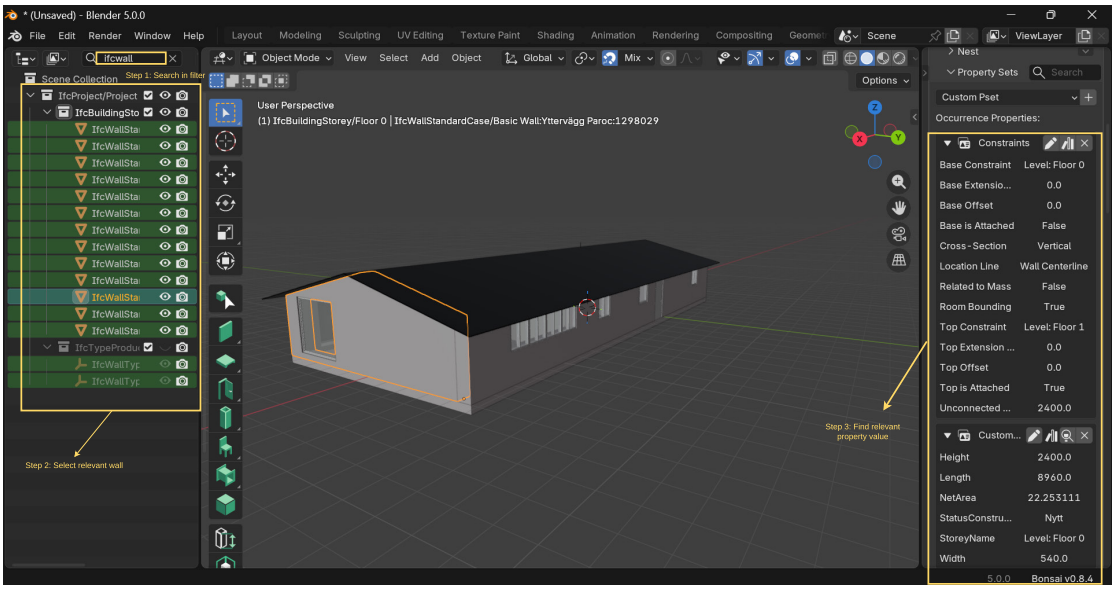


FIGURE 8. Wall/material inspection workflow using Blender/Bonsai.

TABLE 4. Qualitative comparison between conventional BIM inspection tools and BIM-GRASP.

Criterion	Blender/Bonsai	BIM-GRASP
Interaction mode	Graphical inspection	Natural-language QA
Wall/property lookup	Manual selection and panel inspection	Direct query and response
Requires IFC/tool knowledge	Yes	Low
Supports natural-language queries	No	Yes
Supports multi-hop relational queries	Manual and time-consuming	Yes, through graph retrieval
Produces direct natural-language answer	No	Yes

generates a semantically misaligned Cypher query (see Cypher Query column for Q8 in Table 3). Because the LLM hallucinates a generalized `[r:CONNECTED]` relationship rather than querying the exact intermediate IFC relational paths, the system fails to isolate the correct data paths, yielding empty Retriever Results. Consequently, during the Natural Language Translation stage, the framework lacks the necessary relational insights to formulate a response, resulting in the fallback Final Answer: "*I'm sorry, but I do not have information about the types of connections to this wall based on the provided context.*"

C. QUALITATIVE COMPARISON WITH CONVENTIONAL BIM INSPECTION TOOLS

To further contextualize the practical value of BIM-GRASP, we conducted a qualitative comparison with a conventional IFC inspection workflow using Blender with the Bonsai add-on [44], a combination that transforms a standard open-source 3D modeling suite into a fully functional openBIM authoring tool. This comparison was designed to assess how a user would answer representative IFC-related queries using a conventional BIM inspection interface compared to our natural-language question-answering system.

Figure 8 shows an example of the Blender/Bonsai workflow for inspecting wall-related information in our Midsize IFC model used in our experiments. For a simple query such as "*What is the dimension area of this wall in the building?*", the user must perform the three steps shown in the figure consecutively. Step 1, the user must filter the model hierarchy for wall objects. Step 2, They must identify the relevant *IfcWallStandardCase* elements and select individual wall instances. Step 3, they need to manually inspect the corresponding IFC material or property panels. This demonstrates that existing BIM tools can provide access to the underlying IFC data, but the interaction remains manual. It requires familiarity with the tool interface, IFC object naming conventions, and the location of relevant property sets.

As for retrieving answers for multihop questions similar to the ones we are using in our experiment, the user will need to go through a more complicated process. For example, answering "*Which wall has the largest dimension volume in the building?*" requires the user to first identify the selected wall, determine which other walls are connected to it, inspect each connected wall individually, retrieve the relevant dimension volume property values, and manually compare them. Similarly, an input query such as "*Which wall connected to a selected wall has the lowest thermal resistance?*" requires repeated inspection of wall instances and manual comparison of thermal resistance values.

TABLE 5. Categorisation of IFC properties.

IFC Term	Meaning in Natural Language
Connections	
Element_Type	Type of the connected element
Element_ID	Unique identifier of the connected element
Element_Name	Name of the connected element
ConnectionType	Type of connection between elements
Material	
Analytical Properties.Absorptance	Measure of the fraction of solar energy absorbed by the element's surface
Analytical Properties.Heat Transfer Coefficient (U)	Rate of heat transfer per unit area and temperature difference
Analytical Properties.Roughness	Surface texture classification affecting thermal and physical behaviour
Analytical Properties.Thermal Resistance (R)	Measure of the element's resistance to heat flow
Analytical Properties.Thermal mass	Ability of a material to absorb and store heat
Construction.Function	Intended function of the construction assembly
Construction.Width	Thickness of the construction assembly
Construction.Wrapping at Ends	Whether materials are wrapped at the ends of the element
Construction.Wrapping at Inserts	Whether materials are wrapped at inserted components
Materials and Finishes.Structural Material	Structural material used in the element
Finishes.Structural Material	Material finish applied to the structural component
Geometry	
BaseQuantities.GrossFootprintArea	Total horizontal projected area of the element
BaseQuantities.GrossSideArea	Total area of the vertical sides of the element
BaseQuantities.GrossVolume	Total volume enclosed by the element
BaseQuantities.Height	Overall height of the element
BaseQuantities.Length	Overall length of the element
BaseQuantities.Width	Overall width of the element
Dimensions.Area	Measured surface area of the element
Dimensions.Length	Measured length of the element
Dimensions.Volume	Measured volume of the element
Constraints	
Constraints.Base Constraint	Reference level or object that defines the base of the element
Constraints.Base Extension Distance	Additional vertical distance extending from the base
Constraints.Base Offset	Vertical displacement from the base reference
Constraints.Base is Attached	Whether the base is fixed to another element
Constraints.Location Line	Reference line controlling the position of the element
Constraints.Related to Mass	Whether the constraint is linked to a mass object
Constraints.Room Bounding	Whether the element forms a room boundary
Constraints.Top Constraint	Reference level or object that defines the top of the element
Constraints.Top Extension Distance	Additional vertical distance extending from the top
Constraints.Top Offset	Vertical displacement from the top reference
Constraints.Top is Attached	Whether the top is fixed to another element
Constraints.Unconnected Height	Height of the element when it is not connected to another

These examples show that, although Blender/Bonsai can expose IFC properties through its graphical interface, it does not provide direct natural-language querying, automatic graph traversal, or direct answer generation. Table 4 summarizes the comparison between conventional BIM inspection tools and BIM-GRASP. The purpose of this comparison is not to suggest that tools such as Blender/Bonsai are inadequate for BIM inspection. Rather, it clarifies that they are primarily designed for graphical model exploration and property inspection. In contrast, BIM-GRASP is designed to provide direct natural-language access to IFC model information by translating user queries into graph-based retrieval operations and returning natural-language final answers.

VI. CONCLUSION

In this study, we addressed the challenge of parsing and reasoning over the structurally complex IFC schema by developing BIM-GRASP which is a natural language question-answering framework. Our system simplifies access to building models by translating user queries into graph-based retrieval tasks, effectively bridging the gap between intricate IFC data and non-technical stakeholders. By integrating Graph-RAG with a GPT-4.1 generative model optimized with In-Context Learning (ICL), the system achieved an overall accuracy of 88% and significantly reduced false negatives in multi-hop reasoning scenarios by preserving the relational structure of IFC entities.

A critical finding of our research illuminates why standard vector-based RAG architectures perform poorly on IFC datasets. Standard RAG struggles because vital topological information such as how different architectural elements connect is highly fragmented across the file rather than explicitly localized. Standard RAG could theoretically achieve better results if the raw IFC files were manually restructured to include hard-coded, chunkable descriptions of these inter-element connections. However, such manual manipulation is highly labor-intensive and unscalable for complex projects.

Graph-RAG, on the other hand, resolves this problem by inherently connecting elements as nodes and edges in a graph, and hence captures the relational information between IFC elements. Despite these successes, the proposed BIM-GRASP model has several limitations. First, our current

evaluation primarily focused on specific architectural core components, such as *IfcWall* and its immediate spatial properties, to validate the proof-of-concept. The empirical validation of the system is strongest for wall-centered IFC querying and our four selected property categories (Geometry, Constraints, Connections, and Material). The system's performance on highly specialized, non-architectural domains like MEP (Mechanical, Electrical, and Plumbing) systems, which involve entirely different and highly intricate connection schemas, remains untested. Second, relying on an LLM for Cypher query generation can introduce occasional latency and risks semantic misalignment if a user employs highly ambiguous terminology not covered by the ICL prompt library. Finally, the framework requires pre-processing the raw IFC file into a Neo4j graph database prior to querying, a step that can be computationally intensive and time-consuming for exceptionally large, federated commercial models.

Future work will aim to address these limitations by expanding the system's underlying ontology to encompass all IFC classes and to add the BIM-GRASP in an expaneded pipeline for automatic compliance checking. We also plan to explore advanced fine-tuning techniques specifically for Cypher query generation to reduce the reliance on extensive prompt contexts and to improve inference speeds. Finally, we will continue to refine our user interface and software artefact to better support real-world deployment.

APPENDIX
EXAMPLE OF IFC ELEMENTS

See Table 5.

REFERENCES

[1] T. Liebich, Y. Adachi, J. Forester, J. Hyvarinen, K. Karstila, K. Reed, S. Richter, and J. Wix. (2010). *Buildingsmart: Industry Foundation Classes, IFC2x Edition 4 Release Candidate 2. Online*. [Online]. Available: https://technical.buildingsmart.org/standards/ifc/

[2] Z. Xu, T. Huang, B. Li, H. Li, and Q. Li, "Developing an IFC-based database for construction quality evaluation," *Adv. Civil Eng.*, vol. 2018, no. 1, Jan. 2018, Art. no. 3946051.

[3] I. Faraj, M. Alshawi, G. Aouad, T. Child, and J. Underwood, "An industry foundation classes Web-based collaborative construction computer environment: WISPER," *Autom. Construct.*, vol. 10, no. 1, pp. 79–99, Nov. 2000.

[4] R. K. Dhillon, M. Jethwa, and H. S. Rai, "Extracting building data from BIM with IFC," *Int. J. Recent Trends Eng. Technol.*, vol. 11, no. 2, p. 202, 2014.

[5] TNO & Open Source Contributors. (2024). *BIMserver*. Accessed: Mar. 25, 2025. [Online]. Available: https://github.com/opensourceBIM/BIMserver

[6] Open Source BIM Collective. (2024). *BlenderBIM Add-on*. Accessed: Mar. 25, 2025. [Online]. Available: https://blenderbim.org

[7] Q. Zhao, Y. Li, X. Hei, and M. Yang, "A graph-based method for IFC data merging," *Adv. Civil Eng.*, vol. 2020, no. 1, Jan. 2020, Art. no. 8782740.

[8] T. W. Kang and C. H. Hong, "IFC-CityGML LOD mapping automation using multiprocessing-based screen-buffer scanning including mapping rule," *KSCE J. Civil Eng.*, vol. 22, no. 2, pp. 373–383, Feb. 2018.

[9] J. Zhu, P. Wu, and X. Lei, "IFC-graph for facilitating building information access and query," *Autom. Construction*, vol. 148, Apr. 2023, Art. no. 104778.

[10] T. Krijnen and J. Beetz, "A SPARQL query engine for binary-formatted IFC building models," *Autom. Construction*, vol. 95, pp. 46–63, Nov. 2018.

[11] H. Liu, M. Lu, and M. Al-Hussein, "Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry," *Adv. Eng. Informat.*, vol. 30, no. 2, pp. 190–207, Apr. 2016.

[12] X. Du, Y. Duan, and W. Huang, "Attribute information extracting method for air quality assessment of buildings," in *Proc. 7th Annu. Int. Conf. Netw. Inf. Syst. Comput. (ICNISC)*, Jul. 2021, pp. 483–488.

[13] Y. Park and S. Kang, "Natural language generation using dependency tree decoding for spoken dialog systems," *IEEE Access*, vol. 7, pp. 7250–7258, 2019.

[14] N. Wang and R. Issa, "Ontology-based building information model design change visualization," in *Proc. Workshop Linked Building Data Semantic Web Technol. (WLS)*, 2019, pp. 53–61.

[15] S. Wu, Q. Shen, Y. Deng, and J. Cheng, "Natural-language-based intelligent retrieval engine for BIM object database," *Comput. Ind.*, vol. 108, pp. 73–88, Jun. 2019.

[16] J.-R. Lin, Z.-Z. Hu, J.-P. Zhang, and F.-Q. Yu, "A natural-language-based approach to intelligent data retrieval and representation for cloud BIM," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 31, no. 1, pp. 18–33, Jan. 2016.

[17] N. Wang, R. R. A. Issa, and C. J. Anumba, "NLP-based query-answering system for information extraction from building information models," *J. Comput. Civil Eng.*, vol. 36, no. 3, May 2022, Art. no. 04022004.

[18] J. Zhang and N. M. El-Gohary, "Extending building information models semiautomatically using semantic natural language processing techniques," *J. Comput. Civil Eng.*, vol. 30, no. 5, Sep. 2016, Art. no. 4016004.

[19] X. Shi, Y.-S. Liu, G. Gao, M. Gu, and H. Li, "IFCdiff: A content-based automatic comparison approach for IFC files," *Autom. Construction*, vol. 86, pp. 53–68, Feb. 2018.

[20] O. W. Chong and J. Zhang, "Automated extraction of locational information from IFC-based building information models," in *Proc. Construct. Res. Congr.*, 2024, pp. 148–156.

[21] X. Xinglei, M. Zhiliang, H. Huanmin, and X. Peng, "Automated extraction of attributes of IFC objects based on graph theory and SPARQL query," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 1218, no. 1, 2022, Art. no. 012024.

[22] N. Wang and R. R. Issa, "Natural language generation from building information models for intelligent NLP-based information extraction," in *Proc. EG-ICE Workshop Intell. Comput. Eng.* Berlin, Germany: Universitätsverlag der TU Berlin, 2020, pp. 275–284.

[23] R. Zhang and N. El-Gohary, "Transformer-based approach for automated context-aware IFC-regulation semantic information alignment," *Autom. Construction*, vol. 145, Jan. 2023, Art. no. 104540.

[24] J. Zheng and M. Fischer, "BIM-GPT: A prompt-based virtual assistant framework for BIM information retrieval," 2023, *arXiv:2304.09333*.

[25] S. I. Sima Iranmanesh, H. S. Hadeel Saadany, and E. V. Edlira Vakaj, "LLM-assisted graph-RAG information extraction from IFC data," in *Proc. Comput. Construction*, vol. 6, Iran. Porto, Portugal: European Council on Computing in Construction, Jul. 2025, pp. 336–342. [Online]. Available: https://ec-3.org/publications/conference/paper/?id=EC32025_366

[26] International Organization for Standardization, *Industry Foundation Classes (IFC) for Data Sharing in the Construction and Facility Management Industries—Part 1: Data Schema*, Standard ISO 16739-1:2018, Geneva, Switzerland, 2018. [Online]. Available: https://www.iso.org/standard/70303.html

[27] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, and T. Rocktäschel, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 9459–9474.

[28] A. Grattafiori et al., "The llama 3 herd of models," 2024, *arXiv:2407.21783*.

[29] J. Achiam et al., "GPT-4 technical report," 2023, *arXiv:2303.08774*.

[30] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering," *Trans. Assoc. Comput. Linguistics*, vol. 11, pp. 1–17, Jan. 2023.

[31] B. Peng, Y. Zhu, Y. Liu, X. Bo, H. Shi, C. Hong, Y. Zhang, and S. Tang, "Graph retrieval-augmented generation: A survey," 2024, *arXiv:2408.08921*.

[32] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2024, pp. 6491–6501.

[33] H. Huang, Y. Huang, Y. Junjie, Z. Pan, Y. Chen, K. Ma, H. Chen, and J. Cheng, "Retrieval-augmented generation with hierarchical knowledge," in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, C. Christodoulopoulos, T. Chakraborty, C. Rose, and V. Peng, Eds., Suzhou, China: Association for Computational Linguistics, Nov. 2025, pp. 6044–6060. [Online]. Available: https://aclanthology.org/2025.findings-emnlp.321/
[34] M. Busch, G. Defilippis, P. Weiß, C. Beecks, S. Decker, and D. Collarana, "GRAFT–graph retrieval augmented generation fine-tuning approach," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Dec. 2025, pp. 5583–5592.
[35] X. Guan, Y. Liu, H. Lin, Y. Lu, B. He, X. Han, and L. Sun, "Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting," in *Proc. AAAI Conf. Artif. Intell.*, 2024, vol. 38, no. 16, pp. 18126–18134.
[36] S. Afrose, "Retrieval-augmented generation based doctor recommendation system using knowledge graph," Ph.D. dissertation, Dept. Comput. Sci. Eng., Brac University, Dhaka, Bangladesh, 2024.
[37] S. L. Phyu, S. Jaman, M. Uchkempirov, and P. Kulkarni, "Myanmar law cases and proceedings retrieval with GraphRAG," in *Proc. IEEE Int. Conf. Big Data (BigData)*, Dec. 2024, pp. 2506–2513.
[38] J. Wu, J. Zhu, Y. Qi, J. Chen, M. Xu, F. Menolascina, and V. Grau, "Medical graph RAG: Towards safe medical large language model via graph retrieval-augmented generation," 2024, *arXiv:2408.04187*.
[39] Neo4j Inc. (2025). *Neo4j 2025.06.1 Series–graph Database Platform*. Accessed: Aug. 12, 2025. [Online]. Available: https://neo4j.com/release-notes/
[40] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An evolving query language for property graphs," in *Proc. Int. Conf. Manage. Data*, May 2018, pp. 1433–1445.
[41] IFCOpenShell. (2024). *IFCOpenShell 0.8.1*. Accessed: Mar. 25, 2025. [Online]. Available: https://ifcopenshell.org/index.html
[42] Y. Yu, S. Kim, H. Jeon, and B. Koo, "A systematic review of the trends and advances in IFC schema extensions for BIM interoperability," *Appl. Sci.*, vol. 13, no. 23, p. 12560, Nov. 2023.
[43] DeepSeek. (2024). *DeepSeek Coder 6.7b Instruct*. fine-tuned on 2B tokens of instruction data for code generation and completion tasks. [Online]. Available: https://huggingface.co/deepseek-ai/deepseek-coder-6.7b-instruct
[44] *Blender—A 3D Model. Rendering Package*, Blender Online Community, Blender Foundation, Stichting Blender Foundation, Amsterdam, The Netherlands, 2024. [Online]. Available: http://www.blender.org



HADEEL SAADANY is currently a Senior Lecturer in data science and natural language processing (NLP) with Birmingham City University. She conducts research in multidisciplinary fields focusing on the intersecting areas of NLP, deep learning, and artificial intelligence (AI) in different domains. Her research projects are primarily focused on utilizing AI to solve real-life problems through collaborative research with U.K. and USA industrial partners. She has previously worked at the University of Surrey as a Postdoctoral Research Fellow, where she acted as a Lead Researcher in a research-industrial project between eBay and the Institute for People-Centered Artificial Intelligence (PAI). She employed recent large language model AI technology to improve the efficacy of the infrastructure of the search engine for eBay cooperation. She has also been the Lead Researcher of an Innovate U.K. research-industrial project, where she applied NLP techniques for building a multi-modal user interface (UI) system that facilitates legal information acquisition for U.K. Supreme Court. The UI system is currently in the application process to be published as an innovative patent by U.K. Intellectual Property Office.



SIMA IRANMANESH received the M.Sc. degree in software engineering from Yazd University, Iran, and the Ph.D. degree in computer science from the University of Warwick, U.K. She is currently a Lecturer in data science with the School of Computing and Digital Technology, Birmingham City University, U.K. Her research focuses primarily on search algorithms for large-scale graph data, graph-based similarity algorithms, and recommendation systems. She is exploring applications of large language models (LLMs), particularly utilizing retrieval-augmented generation (RAG) and knowledge graph technologies. Additionally, she is an active reviewer for various journals and conferences in the field of data science.



MALIK UZAIR MEHMOOD received the bachelor's degree (Hons.) in information technology and the master's degree in computer science with specialization in natural language processing and machine learning. He is currently pursuing the Ph.D. degree in computing and the built environment with Birmingham City University, U.K. He has more than ten years of combined experience in academia and industry. His research interests include large language models, machine learning, artificial intelligence, and sustainable building design.



EDLIRA VAKAJ is currently a Professor of neuro-symbolic AI with Birmingham City University. She is leading the Research and Innovation at the College of Computing. She conducts research in multidisciplinary projects focusing on semantic web, knowledge graphs, AI, and semantic data spaces. She is the Principal Investigator of the ACCORD Horizon Project and engaged in several European and U.K.-funded projects of various domains where semantic web technologies are applied, such as renewable energy (FP7 RENESENG), industrialized construction and Industry 4.0 (Innovate U.K. DfMA and Knowledge Transfer Partnership), and higher education and youth (Erasmus + Capacity Building, Learning Mobility of Individuals, Cooperation for Innovation and the Exchange of Good Practices Action). She is acting as the Academic Lead of several knowledge transfer partnership (KTP) projects, such as the 4Net KTP and Hadley Group KTP. She is a Professional Member of the British Computer Society; a fellow and AURORA of the Higher Education Academy, U.K.; and a Marie Curie Experienced Researcher Fellow of the University of Surrey. She is an Active Member of various communities, such as the Alan Turing Knowledge Graph Community, Linked Building Data, Knowledge, Graph Creation, and Common Action.



EDDIE KASSAEI received the Bachelor of Science degree in architecture from Iran University of Science and Technology and the Master's of Architecture degree from the University of Houston. He is currently the Director of product with Inertia Systems, where he focuses on construction technology, BIM workflows, cloud-based collaboration, and data-driven product development for the AECO industry. He has more than a decade of combined experience across architecture, BIM, and enterprise construction software, with previous product leadership roles at Procore Technologies, Autodesk, and Assemble Systems. His work focuses on making complex building and project data more accessible to construction stakeholders through intuitive workflows, model-based data extraction, knowledge graphs, and AI-assisted product experiences.