Data & Knowledge Engineering 164 (2026) 102581

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

ASK-BIM: A knowledge graph-powered AI system for natural language querying of BIM models☆

Andrea Ibba [a,b], Rubén Alonso [b,c], Diego Reforgiato Recupero [a,b,*]

[a] Department of Mathematics and Computer Science, Università degli Studi di Cagliari, Via Ospedale 72, Cagliari, 09124, Italy
[b] ICT Division, R2M Solution s.r.l., 42, Pavia, 27100, Italy
[c] Programa de Doctorado, Centro de Automática y Robótica, Universidad Politécnica de Madrid-CSIC, Madrid, Spain

ARTICLE INFO

Keywords:
Building Information Modeling
Large Language Models
Artificial Intelligence
Knowledge Graphs
Semantic Web
Retrieval-augmented generation

ABSTRACT

Building Information Modeling (BIM) has transformed the Architecture, Engineering, Construction, and Operation (AECO) industry by integrating diverse types of information into a unified digital model. While BIM enhances collaboration and decision, making throughout a project's lifecycle, querying and extracting meaningful insights from BIM models remains a challenge due to their complexity and the technical expertise required to navigate formats like Industry Foundation Classes (IFC). Existing tools provide limited accessibility, and applying state-of-the-art Large Language Models (LLMs) directly to IFC files has proven ineffective due to data volume, lack of semantic structure, and relational complexity.

To address these challenges, we introduce ASK-BIM, an approach that combines LLMs, linked data, and knowledge graph (KG) technologies to enable natural language querying of IFC files. By structuring BIM data into a KG before engaging an LLM for reasoning and query resolution, ASK-BIM enhances data accessibility while preserving semantic relationships crucial for complex queries.

We evaluate ASK-BIM on a real-world multi-storey building, categorizing questions along two axes and assessing performance in extracting relevant information. Our findings demonstrate the potential of graph-based representations to facilitate AI-driven BIM analysis while also identifying challenges related to the extraction of information from the graph structures. By bridging the gap between BIM data and AI reasoning, ASK-BIM represents a significant step toward intuitive and efficient BIM querying through natural language.

1. Introduction

Building Information Modeling (BIM) is a transformative process in the architecture, engineering, and construction and operation (AECO) industry, leveraging digital technologies to create and manage comprehensive project information throughout its lifecycle [1]. At its core, BIM revolves around the development of a detailed digital representation of a physical structure, encompassing geometric, spatial, and informational data. This model serves as a shared resource for construction stakeholders, including designers, engineers, contractors, and facility managers, facilitating collaboration, reducing errors, and improving efficiency. BIM is not limited to design and construction phases but extends into operations, maintenance, renovation or even demolition. By integrating data-driven decision-making and advanced analytical tools, BIM fosters innovation, enhances sustainability, and drives better outcomes

☆ This article is part of a Special issue entitled: 'LLMs and KGs for Semantics-driven Sys Engineering - Woo' published in Data & Knowledge Engineering.

* Corresponding author at: Department of Mathematics and Computer Science, Università degli Studi di Cagliari, Via Ospedale 72, Cagliari, 09124, Italy.
E-mail addresses: andrea.ibba.uni@gmail.com (A. Ibba), ruben.alonso@r2msolution.com (R. Alonso), diego.reforgiato@unica.it (D. Reforgiato Recupero).

https://doi.org/10.1016/j.datak.2026.102581
Received 15 February 2025; Received in revised form 19 December 2025; Accepted 12 February 2026
Available online 19 February 2026

for projects of all sizes and complexities [2]. BIM has thus revolutionized the AECO industry by enabling the integration of diverse types of information into a single digital model. BIM dimensions, ranging from fundamental representations to advanced analytical capabilities, organize this information into structured levels [3]. By extending beyond the geometric 3D representation to include aspects such as time, cost, sustainability and safety, BIM dimensions provide a comprehensive framework for addressing the complexities of modern construction projects expanding the utility of BIM throughout the lifecycle of a construction project [4].

The first three dimensions establish the foundational framework of a BIM model. The first dimension (1D) encompasses essential textual and numerical data, such as project specifications, codes and requirements. The second dimension (2D) builds on this foundation by incorporating traditional flat representations like drawings, plans, and schematics that guide design and construction. The third dimension (3D) introduces geometric representation, providing a detailed spatial model of the project that enables visualization, coordination and basic analyses. Beyond these foundational levels, BIM dimensions extend into more specialized areas of information. For instance, 4D represents temporal information and time scheduling, while 5D focuses on cost estimation and analysis. Despite BIM's capacity to incorporate n-dimensional information, there is no universal consensus on what each dimension beyond 5D should include. For example, sustainability and energy analysis may be categorized as either 6D or 7D, and safety considerations appear across various levels, ranging from 6D to 8D [5]. Usually, dimensions beyond 5D often encompass aspects such as operations and maintenance, lean construction requirements, and industrialization processes.

Despite its many advantages, querying BIM models presents significant challenges due to their complex and vast amount of data distributed across multiple dimensions. BIM models integrate diverse types of information, including geometric, semantic, and relational data, stored in formats like Industry Foundation Classes (IFC) or proprietary standards [6]. Extracting meaningful insights from these models often requires specialized tools and software capable of navigating their structures. Tools such as Autodesk Revit,[1] Navisworks,[2] or Solibri[3] are commonly used for querying and analyzing, but their effective use demands expertise. Professionals must possess not only technical proficiency in these tools but also a deep understanding of the data's context within the construction lifecycle. This reliance on expert knowledge underscores the importance of specialized training and experience in managing and utilizing BIM data effectively.

Despite the promise of natural language models in enhancing accessibility to BIM data, directly applying state-of-the-art Large Language Models (LLMs) to IFC files remains a complex challenge. While models behind ChatGPT[4] and DeepSeek[5] can process textual representations of IFC files, their effectiveness is significantly constrained by the sheer size and complexity of the data. For instance, DeepSeek, when applied to a 10 MB IFC file, can only analyze approximately 1% of its content, limiting its ability to provide comprehensive responses. Other models, such as GPT-4o, can process the entire file but struggle to generate meaningful answers beyond simple element counts, as IFC data is often represented as structured text rather than semantically connected information. For example, while GPT-4o can determine the total number of walls in an IFC file, it fails to retrieve or reason about their associated properties, such as their respective levels or dimensions. This limitation arises because the IFC data, when converted into plain text, lose their relational structure, preventing the model from effectively linking entities to their attributes and extracting meaningful insights beyond basic counts. The inability of these models to reason over relationships, derive properties, or infer spatial dependencies from the raw text highlights the need for a more structured approach.

Recent research suggests that graph-based representations hold significant promise due to their ability to preserve object relationships and semantic richness, making them highly compatible with advanced AI technologies [7]. By selectively extracting and organizing relevant information into a knowledge graph (KG) before engaging an LLM for reasoning and query resolution, it becomes possible to leverage LLMs effectively while overcoming their limitations in handling complex BIM data.

Among these solutions, ifcOWL provides a Web Ontology Language (OWL) representation of the IFC schema. However, ifcOWL was primarily an academic experiment, which contains numerous exceptions and particularities that make it challenging to use in practice. For this reason, in our work we rely on the IFC-to-LBD Converter, which, as detailed in Section 3.3, first converts the IFC file into its ifcOWL representation and then traverses and reorganizes this representation to produce a more coherent and structured Linked Building Data (LBD) graph that better captures the semantics of the original IFC model.

Thus, to address the challenges of querying BIM models, we propose ASK-BIM, an innovative approach that leverages LLMs, Chain of Thought inspired reasoning strategies, linked data, and KG technologies.

Our approach simplifies data access and enhances decision-making across all stages of a construction project. This method not only reduces the reliance on domain-specific knowledge but also accelerates the process of extracting actionable insights from BIM models.

Following the Design Science Research (DSR) paradigm [8], ASK-BIM is conceived as a design artifact that responds to the need for intuitive access to BIM data. The design process is guided by specific requirements derived from the analysis of current limitations in BIM querying tools and evaluated through a set of user-inspired questions and real-world BIM data.

In accordance with the DSR framework, the scientific objectives of this research can be formulated as the following research questions (RQs):

- **RQ1:** How can KG representations enhance the semantic accessibility of BIM models for natural language querying?

[1] https://www.autodesk.com/products/revit/
[2] https://www.autodesk.com/products/navisworks/
[3] https://www.solibri.com/
[4] https://openai.com/chatgpt/overview/
[5] https://www.deepseek.com

- **RQ2:** How can LLMs be effectively guided to translate natural language queries into structured graph queries (e.g., SPARQL) for BIM data?
- **RQ3:** To what extent does the integration of KGs and LLMs improve the accuracy and usability of BIM querying compared to existing approaches?

Accordingly, the contributions we bring in this paper are the following:

- we present the pipeline of ASK-BIM, a system designed to process natural language questions and retrieve relevant information from IFC files;
- the approach leverages LLMs: prompt engineering techniques are employed to interpret input text and generate SPARQL queries for data retrieval. Then, reasoning capabilities are utilized to provide a final answer;
- to evaluate ASK-BIM, we created a set of 28 questions by consulting professionals and stakeholders in the domain. These questions were categorized along two axes, with a detailed discussion provided for each category;
- ASK-BIM was tested on a complex BIM file representing a real-world multi-storey building in Barcelona. The system successfully processed a wide range of natural language questions and retrieved relevant information, demonstrating its potential to enhance data accessibility throughout the construction lifecycle;
- we identify and analyze key challenges in converting IFC files into graph representations, highlighting the lack of standardization in conversion methods and the resulting inconsistencies across different tools. Additionally, we assess the limitations of LLMs in interpreting complex relationships within graph-based BIM data, which can impact the accuracy of query responses.

This research is framed within activities related to Digital Building Logbooks (DBL) in the innovation projects CHRONICLE[6] and LEGOFIT.[7] DBLs are trusted archived of relevant information throughout a building's lifecycle [9]. These logbooks are directly linked to BIM models, with much of the information included in these models serving as primary sources for the data archived in the DBLs. CHRONICLE is an EU project focused on digitalization and dynamic logbooks for building performance. LEGOFIT, on the other hand, is another EU project that focuses on the construction and renovation of Energy Positive Homes, linking DBLs with BIM models and information about materials and recyclability. Both projects offer an environment to research and validate approaches related to extracting information from BIM models and complementing this information using LLMs.

The remainder of this paper is organized as follows. Section 2 reviews related work on tools and approaches designed to address natural language questions. Section 3 provides the necessary background, including an overview of open BIM concepts, the building utilized in this study, and the tool employed to convert IFC files to RDF format. The pipeline of our proposed approach for leveraging LLMs to respond to natural language questions is detailed in Section 4. The methodology used to construct our dataset of questions and answers is explained in Section 5. Section 6 presents the experimental methodology whereas Section 7 shows the results. Finally, Section 8 concludes the paper with a summary of our findings and a discussion of future research directions.

2. Related works

In the context of Natural Language Processing (NLP), semantic enrichment of BIM focuses on enhancing building models by incorporating meaningful semantics to improve their utility and functionality. Although it holds great potential for advancing BIM technology, semantic enrichment is still an emerging field with an incomplete body of knowledge and lacks formal definitions of its processes, applications, and computational approaches. The study in [10] conducts a comprehensive literature review and bibliometric analysis to identify existing methods and applications of semantic enrichment, particularly in relation to the IFC schema and web ontologies.

The work performed in [11] reviews the application of NLP in the construction industry under the Industry 4.0 paradigm, addressing a gap in comprehensive evaluations of its use in construction-related areas. By analyzing 91 research articles using CiteSpace and VOSViewer, the review highlights key datasets, technologies, tools, applications, and progress in the field. Findings reveal significant challenges, including data isolation leading to non-reproducible research. The review emphasizes the need for interdisciplinary approaches, integrating cross-modal frameworks and end-to-end pre-trained neural network models, to meet future industry demands.

In line with the advancements of Industry 4.0, the work in [12] developed a query-answering (QA) system using NLP to create a virtual assistant for construction project teams. The developed system was evaluated with seven BIM/IFC models and 127 test queries achieving high performances. Other researchers have proposed in [13] an Artificial Intelligence-based framework to streamline access to information within BIM models. The framework begins by using a support vector machine (SVM) algorithm to classify the user's question type. Concurrently, NLP is employed for syntactic analysis to identify the key terms in the user's query. Next, an ontology database, along with latent semantic analysis (LSA), is utilized to achieve a deeper semantic understanding of the question. The identified keywords are expanded based on their semantic relationships within the ontology, and a final query is generated using both the original keywords and their expanded concepts. To retrieve and display the relevant information from BIM models, a Navisworks API is developed, which leverages the identified question type and parameters to extract the necessary results for the user.

[6] https://doi.org/10.3030/101069722
[7] https://doi.org/10.3030/101104058

Another work performed in [14] proposed an NLP-based approach for intelligent data retrieval and representation in cloud BIM applications which introduced keywords and constraints to capture user requirements. These last are mapped to IFC entities and properties using the International Framework for Dictionaries (IFD). A graph-based pathfinding approach connects user requirements to the IFC schema for effective data retrieval and analysis.

One more study performed in [15] addresses the challenge of retrieving information from BIMs using natural language interface systems. While existing systems struggle to handle complex, constraint-based queries, this work introduces an ontology-aided semantic parser to bridge this gap. The approach maps natural language queries with multiple constraints into computer-readable codes for BIM data retrieval. A modular ontology was developed to represent IFC concepts, relationships, and reasoning rules, incorporating project-specific details. The parser extracts concepts and constraints from natural language queries to generate SPARQL queries for precise BIM retrieval. The system achieved a 91% accuracy rate on a set of 225 natural language queries. Other researchers in [16] introduced a conversational AI system to facilitate attribute information retrieval from BIM using advanced deep learning techniques. The system leverages Google's pre-trained Bidirectional Encoder Representations from Transformers (BERT) language neural network and natural language generation methods, implemented in a Python-based prototype. Testing showed the system achieved an 80% accuracy in providing precise answers to building attribute queries.

More recently, LLM-based methods have been proposed to generate SPARQL queries from natural language inputs. In [17], the authors present an LLM architecture that combines schema selection and entity combination strategies to generate SPARQL queries from user questions over large-scale knowledge bases. They reduce the set of candidate schema elements (classes and relations) shown to the LLM, and generate multiple candidate queries by combining different entity or linking options, thereby improving both efficiency and accuracy. While their system demonstrates competitive benchmark performance, it remains focused on general-purpose knowledge bases and does not explicitly address highly domain-specific ontologies or spatial and relational modeling as found in BIM or IFC datasets.

Similarly, the work in [18] applies a retrieval-augmented generation (RAG) approach to the bioinformatics domain, where the LLM generates federated SPARQL queries across multiple KGs. Their system makes use of metadata, such as example question-query pairs and VoID descriptions of endpoints, and includes a post-generation validation step to check that the generated query aligns with the schema of the target endpoints. This approach excels in handling federated endpoints and domain-specific data but is still geared toward large life-science KGs rather than building information models or semantic representations of BIM data.

These approaches illustrate the growing relevance of LLM-based query generation pipelines and highlight key design strategies, such as schema filtering, context retrieval, multi-candidate query generation, and validation, that inform the system proposed in this paper.

Our work differs from these approaches in that it targets the BIM and IFC domain rather than general-purpose or bioinformatics KGs. It exploits ontologies specifically designed for building models, incorporating spatial hierarchies, element containment, and relationships between storeys and spaces. In addition, it integrates a traversal-based converter step as part of the pipeline to ensure that the graph structure faithfully represents the semantics of the original model. Unlike existing systems, the proposed approach leverages prompt engineering techniques of LLMs in combination with established building ontologies and Semantic Web best practices. This enables it to efficiently interpret and respond to natural language questions about BIM data, providing a more robust, context-aware, and accurate query-answering capability tailored to the built environment domain. Moreover, our approach not only enhances the accessibility of BIM information but also paves the way for more intelligent, scalable, and user-friendly solutions in the field.

One promising direction, for example, involves the integration of BIM models with IoT sensor data [19] to facilitate the development of multidimensional digital twins. Such twins can incorporate real-time measurement data alongside spatial information, while leveraging the multiple dimensions of BIM models to include scheduling, circularity, and safety aspects. An approach like the one proposed in this work would allow users to query and retrieve information from this BIM-based digital twin of the physical environment, effectively exploiting the implicit knowledge encoded within LLMs.

3. Background and methods

3.1. Open BIM

BIM models can be classified as either open or closed, depending on the data storage format. Open formats (e.g., IFC) enhance interoperability and data compatibility, making them more suitable for programmatic processing and conversion into formats compatible with LLMs. Closed formats, on the other hand, are specific to a particular software or company (e.g., RVT, Autodesk Revit). While they may support more advanced capabilities, they are usually only operable within tools from the same provider.

While it is true that the use of proprietary tools and closed formats remains common, there is an increasing interest in Open BIM concepts and open, interoperable standards that enable the processing and modification of BIM models across multiple applications within the construction software ecosystem. Open BIM can be defined as a universal approach based on open workflows and data, for the collaborative design, construction, and operation of buildings [20]. This universality facilitates the development of innovative applications and, above all, integration with solutions and approaches that are not necessarily designed for the construction industry.

Within Open BIM, IFC is the most widely used format. IFC defines an open data schema that enables the description of information related to a building [21], including its geometry and information across the BIM dimensions mentioned in Section 1.

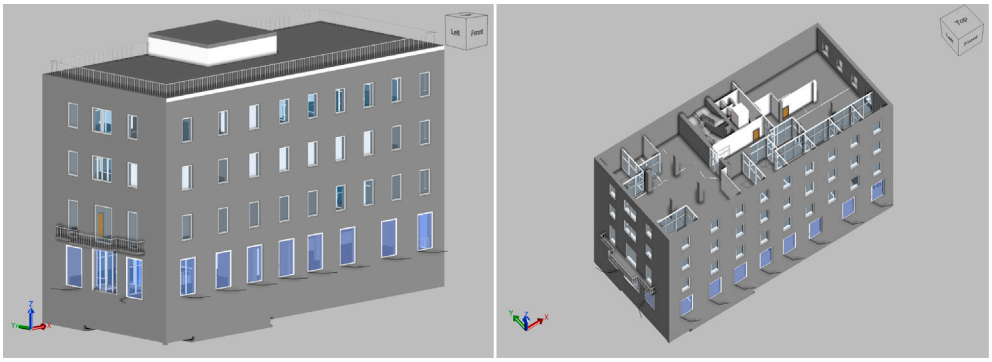


Fig. 1. IFC Model of the analyzed Barcelona office building.

IFC is supported by various tools and schemas that enhance processing and interoperability. For example, universal communication protocols like the BIM Collaboration Format (BCF)[8] and requirement definition standards such as the Information Delivery Specification (IDS)[9] play a crucial role in streamlining data exchange and coordination.

Outside of the IFC ecosystem, within open schema frameworks, it is also important to mention Green Building XML (gbXML),[10] an open schema focused on energy modeling and software interoperability for construction projects. Green Building XML serves as a common format for exporting data from multiple commercial software platforms, thus acting as a bridge between different solutions, a method of data exchange, and occasionally as a complement to the IFC model.

3.2. The analyzed building

In the case of this work, IFC models with geometric information and spatial relationships between elements, have been used. The open schema facilitates processing, information extraction, and interoperability.

The models detailed in this section are derived from Computer-aided design to BIM (CAD2BIM) and photogrammetry or LiDAR scanning to BIM (Scan2BIM) processes. In some cases, the process begins with building plans and other information defined in Computer-aided design models that are manually converted to BIM by a team of experts. This is a common practice in the construction industry and provides initial BIM models that can be enriched with additional data (e.g., material information or time scheduling). For Scan2BIM cases, the process starts with photogrammetric data or information obtained through LiDAR, which is used to infer three-dimensional geometries. These geometries are then modeled and enhanced with added data and metadata. In both scenarios, the results are exported as IFC format BIM models that are utilized within the ASK-BIM workflow.

Specifically, the primary sources are BIM models created in various projects and commercial activities by R2M's Innovative Products division as part of their Scan2BIM service. For example, the validation presented in this work has been carried out on an IFC file representing an office building in Barcelona (Spain). It represents 4 floors of offices and is shown in Fig. 1.

3.3. IFC to knowledge graph converters

IFC files provide a comprehensive representation of building models, encompassing spatial relationships, material properties, dimensions, and semantic metadata. This richness makes the format ideal for capturing and sharing building information among various stakeholders. Currently, several methods exist to programmatically extract information from IFC files, such as IfcOpenShell,[11] or ifcOWL.[12] However, such toolkits require expertise in both BIM knowledge and programming, making the process challenging for non-expert users. To address this barrier, researchers have recently begun developing solutions that allow common users to access and utilize BIM data without specialized knowledge.

The IFC-to-LBD[13] converter, which we adopted within our pipeline, offers a more streamlined and usable alternative [22]. Its modular design and adherence to World Wide Web Consortium (W3C) standards make it well-suited for enabling semantic reasoning and efficient querying in building-related Linked Data applications.

It takes as input an IFC file, and first generates its ifcOWL representation, which expresses the IFC schema using the Web Ontology Language (OWL). In the next stage, the converter systematically traverses this intermediate graph, following hierarchical and relational paths such as those linking sites, buildings, storeys, spaces, and contained or adjacent elements. During this traversal,

[8] https://technical.buildingsmart.org/standards/bcf/
[9] https://www.buildingsmart.org/what-is-information-delivery-specification-ids/
[10] https://www.gbxml.org/
[11] https://ifcopenshell.org/
[12] https://technical.buildingsmart.org/standards/ifc/ifc-formats/ifcowl/
[13] https://github.com/jyrkioraskari/IFCtoLBD

the converter extracts and reorganizes entities, properties, and relationships, mapping them to the corresponding classes and properties adhering to the LBD schema.[14] The LBD schema is based on a set of standardized ontologies developed by the W3C Linked Building Data Community Group.[15] The structure of the LBD graphs is more aligned with a Semantic Web environment, achieving a minimum reduction of 83% in the number of triples compared to the IFC OWL graph generated from the same IFC file from the ifcOWL tool previously introduced. The converter performs the transformation based on three LBD ontologies, ensuring a reusable and modular representation of building data while maintaining compliance with Semantic Web principles. The key ontologies involved include:

- **Building Topology Ontology (BOT)** [23][16]: Designed to represent the topology of buildings, such as sites, buildings, storeys, spaces, and elements. BOT adheres to Linked Data best practices and facilitates semantic clarity and interoperability. It also includes the **Building Element Ontology (BEO),**[17] which is derived from the `IfcBuildingElement` subtree in the IFC specification.
- **Building-Related Properties Ontology (PROPS)**[18]: Designed to represent the properties of building elements, supporting different levels of complexity, from simple properties to grouped and stateful properties. This modularity enables flexible data linking and extensibility.
- **Product Ontology (PRODUCT)**[19]: Focused on classifying building elements (e.g., walls, doors, and windows) and integrating these classifications with building topology and property ontologies.

In the following, we briefly introduce the main classes and relationships defined in the ontologies adopted in this work, which together provide the semantic foundation for representing building structure, components, and properties within the generated KG. The *BOT* ontology defines the spatial hierarchy of the built environment through the concept of `bot:Zone`, representing any portion of physical or virtual space with a three-dimensional extent. BOT follows a hierarchical "Matryoshka doll" structure, in which a `bot:Site` contains one or more `bot:Building` instances (linked via `bot:hasBuilding`), each composed of one or more `bot:Storey` elements (`bot:hasStorey`), which in turn contain `bot:Space` instances (`bot:hasSpace`). Adjacent spaces are connected through the `bot:adjacentZone` relationship, while the transitive property `bot:containsZone` generalizes these hierarchical relations, enabling queries that traverse from site to individual rooms. This structure provides the foundation for representing the physical composition and spatial relationships within a building.

Complementing BOT, the *BEO* ontology extends the model with classes describing tangible construction elements. For example, a `beo:Door` represents a complete unit composed of a frame and one or more leaves, including its hardware and weatherseal, and is defined as a subclass of the more generic `beo:BuildingElement`. BEO also includes subclasses such as `beo:Gate` and `beo:RoofHatch`, supporting finer granularity in representing specific building components. Used together, BOT and BEO enable the representation of both spatial hierarchy and physical components, providing a semantically rich foundation for subsequent reasoning and querying over BIM-derived graphs.

The *PRODUCT* ontology complements BOT and BEO by describing the tangible components of the built environment and their spatial associations within the building topology. While BOT focuses on spatial containment (`bot:hasSpace`, `bot:hasStorey`) and BEO specifies element types (e.g., `beo:Door`, `beo:Wall`), PRODUCT provides a bridge between spatial zones and physical products. For instance, a `bot:Space` can contain or adjoin building elements such as a `product:Heater`, `product:Wall`, or `product:Floor` through relations like `bot:containsElement` and `bot:adjacentElement`.

Finally, the *PROPS* ontology structures the property information associated with building elements. It provides a flexible framework to represent element attributes at varying levels of granularity, ranging from simple literal assignments to fully versioned property instances with temporal tracking. The specific representation levels supported by PROPS are detailed later in this section, illustrating how properties can be progressively enriched to support advanced querying and historical data management. Together, BOT, BEO, PRODUCT, and PROPS form an interconnected ontology ecosystem that supports both spatial and semantic reasoning over BIM data, ensuring consistency, interoperability, and scalability.

The resulting KG organizes, therefore, the building data into nodes (representing classes and instances) and edges (representing relationships or properties). For instance, an IFC class such as "IfcWall" is represented in the KG as a class `beo:Wall`, with properties like `props:area_property_simple` or `props:level_property_simple` and relationships with other nodes. This structure allows the data to be queried efficiently using SPARQL.

As an example, during the conversion, the following extract of the referenced IFC file has been traversed.

#98535=IFCWALLSTANDARDCASE('33DqYC4r15jApJ0RLMJdB2',#18,'Basic Wall:Interior - 79 mm Partition (1-hr):451442',$, 'Basic Wall:Interior - 79 mm Partition (1-hr)',#98512,#98534,'451442');

From this traversal, relevant classes and properties are identified and extracted, contributing to the construction of the corresponding `beo:Wall` RDF entity.

Fig. 2 presents the `beo:Wall` instance generated through the conversion process by traversing the previously mentioned extract of the referenced IFC file for the selected building. It includes relevant properties derived from this traversal and incorporated into the final representation.

[14] https://w3c-lbd-cg.github.io/lbd/UseCasesAndRequirements/
[15] https://github.com/w3c-lbd-cg/
[16] https://w3c-lbd-cg.github.io/bot/
[17] https://pi.pauwel.be/voc/buildingelement#
[18] https://github.com/maximelefrancois86/propsl
[19] https://github.com/w3c-lbd-cg/product

```
inst:wall_c337488c-1350-45b4-acd3-01b5564e72c2
rdf:type    beo:Wall , bot:Element;
rdfs:label  "Basic Wall:Interior - 79mm Partition (1-hr):451442";
props:type_property_simple  "Interior - 79mm Partition (1-hr)";
owl:sameAs  inst:IfcWallStandardCase_98535;
```

Fig. 2. Wall instance.

Fig. 3. Graph excerpt of the `beo:Wall` instance visualized in GraphDB, showing its associated entities and relationships.

To visually illustrate how this instance and its related entities are represented within the KG, Fig. 3 shows an excerpt from the GraphDB visualization. The resulting subgraph highlights the `beo:Wall` instance, its RDF types, and linked properties and relationships, providing a clear view of how IFC entities are transformed into interconnected nodes and edges within the Linked Building Data model.

The IFC-to-LBD conversion occurs in two phases: first, the IFC file is converted to a temporary IFC OWL graph using an IFC-to-RDF converter, and subsequently, this intermediate graph is incrementally converted into an LBD graph. However, this simplification introduces certain limitations. For instance, some material properties and geometric details, such as non-standard attributes, may be lost during the conversion.

The converter supports various IFC file versions, including IFC2x3 TC1 and IFC4 Add2. It allows users to convert an IFC building model into a single RDF file combining all selected LBD modules (BOT, PROPS, and/or PRODUCT).

During the IFC-to-LBD conversion process, users can select the most appropriate PROPS complexity level for their project, determining how element properties are structured and represented in the resulting KG. The PROPS framework provides three levels of complexity, each offering a different approach to managing properties. At **Level 1 (L1)**, the simplest representation, properties are assigned directly as literal values without the introduction of intermediate instances. If the property value is literal, the generated predicate URI is suffixed with `_simple` to differentiate it from representations at higher levels. At **Level 2 (L2)**, an intermediate representation is introduced to structure properties through separate instances rather than assigning values directly. Instead of storing a property as a simple literal, an intermediate instance is created to encapsulate the value, facilitating richer relationships between properties. This structure enhances flexibility and enables the grouping of properties while maintaining compatibility with simpler representations from L1 when no intermediate structure is required. At **Level 3 (L3)**, the most complex level, additional layers of information are incorporated, introducing property state tracking and enabling versioning. At this level, beyond the

intermediate instance representing the property, a dedicated instance is also created to store the state of the property, allowing for temporal tracking and historical data management. By using PROPS L3, it becomes possible to implement property versioning within IFC-based building models with minimal additional implementation effort. This structured approach facilitates tracking property changes over time, improving the ability to manage historical data and enabling more advanced analysis in BIM environments. The progressive nature of these levels ensures that complexity is introduced only where necessary. L2 and L3 do not replace the representations of previous levels but extend them, allowing for richer semantic relationships and enhanced query capabilities while preserving compatibility with simpler data structures. By selecting the appropriate PROPS level, users can tailor the conversion output to match their specific data representation needs, balancing simplicity and semantic richness based on project requirements. The progressive nature of these levels ensures that complexity is introduced only where needed, maintaining compatibility with less detailed representations while enabling more advanced semantic relationships.

4. The proposed pipeline

The pipeline developed for ASK-BIM is depicted in Fig. 4. It takes as input an IFC file representing a construction (e.g., a building, infrastructure, or industrial facility) along with a user question in natural language. The ASK-BIM tool transforms the IFC file into a queryable KG, generating a natural language answer to the input question. This is accomplished through Chain of Thought-inspired prompting strategies [24] that generate SPARQL queries and reasoning steps, ensuring context-aware and accurate responses.

The pipeline is structured into three interconnected blocks:

- **IFC File to Knowledge Graph Conversion:** This block transforms the IFC file into a KG enabling efficient data organization and queryability.
- **User Question Processing and SPARQL Query Generation:** In this block, the user's question is analyzed and mapped to relevant entities, properties, and relationships within the KG. By using an LLM, SPARQL queries are generated using prompt-based techniques to retrieve the necessary data.
- **Aggregation and Final Answer Generation:** In the final step, the LLM's reasoning capabilities are used with the results of the SPARQL queries to generate a concise, user-friendly response in natural language, precisely tailored to the user's query.

The full source code of the pipeline, including the scripts for IFC conversion, KG ingestion, prompting workflows, SPARQL query generation, and answer synthesis, is openly available on GitHub.[20] The repository provides a complete, executable version of the ASK-BIM framework, implemented primarily in Python, with additional shell scripts for managing conversion workflows and configuration files for integrating with the GraphDB triplestore. The pipeline can be triggered on demand by running the main entry-point script once the required environment is set up. Specifically, the execution requires a running triplestore instance (such as GraphDB) containing an IFC-based KG and a valid OpenAI API key to enable NLP. The repository also includes example datasets, configuration files, and prompt templates used during the evaluation phase, allowing users to reproduce the experiments and extend the framework for their own research.

To ensure methodological rigor, the prompting strategy adopted in ASK-BIM was systematized using the TELeR taxonomy [25], which characterizes prompts along four dimensions: *Turn*, *Expression*, *Level of detail*, and *Role*.

In ASK-BIM, these dimensions are instantiated as follows. The pipeline employs multi-turn prompting (*Turn*) to decompose complex questions into sub-questions and to progressively build the context that supports SPARQL query generation. Prompts are expressed in natural language with embedded formal constraints (*Expression*) to bridge user intent and graph query syntax. Different stages of the pipeline require varying granularity of information (*Level of detail*), ranging from high-level entity identification to fine-grained property retrieval. Finally, prompts explicitly assign roles (*Role*) to the LLM (e.g., schema analyst, SPARQL generator, or answer synthesizer) to guide reasoning behavior.

This structure ensures that each prompt corresponds to a distinct reasoning function within the system, enabling modular design, explainability, and alignment with established prompting theory.

4.1. IFC file to knowledge graph conversion

The first block of the ASK-BIM pipeline focuses on converting the input IFC file into a KG. Our input IFC file, a multi-storey building located in the province of Barcelona, contains elements such as stairs, walls, doors, and floors.

In the conversion process, we selected the most complex and information, rich PROPS level (PROPS L3) to maximize the amount of data extracted from the IFC file. The generated KG is then stored in Turtle (.ttl) format, a compact and readable RDF serialization. This format is then imported into a graph database, which in our case is GraphDB.[21] GraphDB acts as the SPARQL endpoint and we chose it for its lightweight architecture, ease of deployment, and compatibility with RDF data, making it well-suited for BIM-related use cases. Then, we executed a SPARQL query to retrieve all the distinct classes from the inferred KG. This set serves as a context for the subsequent steps.

[20] https://github.com/aibba19/ASK-BIM
[21] https://graphdb.ontotext.com/

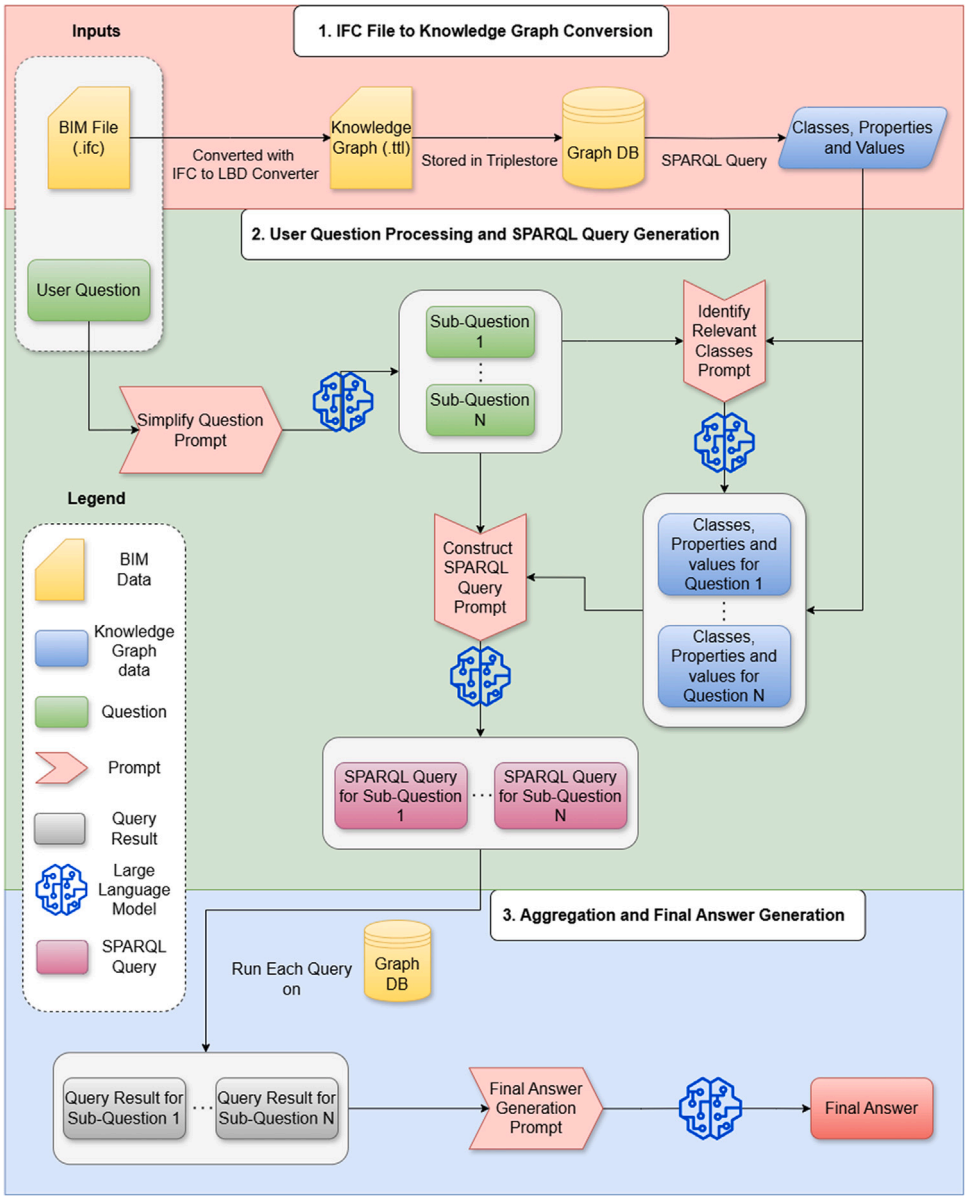


Fig. 4. Overview of the proposed pipeline of ASK-BIM for the question-answering system over BIM data.

4.2. User question processing and SPARQL query generation

The second block of the pipeline focuses on processing the user's input question to identify relevant entities and generate appropriate SPARQL queries. This phase leverages the capabilities of an LLM, specifically ChatGPT, chosen for its state-of-the-art performance and leading position in the market. Prompt engineering techniques are employed to ensure precise entity recognition and accurate SPARQL query formulation.

We adopt a Chain of Thought inspired strategy to enhance reasoning over complex queries. Following Chain of Thought principles, our approach breaks down the problem into a series of intermediate reasoning steps before arriving at a final conclusion. Specifically, rather than directly seeking an answer to each query, we implement a pipeline of intermediate steps to simplify the question, identify relevant KG classes and finally generate the SPARQL query.

We employed three distinct prompts, each addressing a different step in this block. The first prompt simplifies the user's question, breaking it into independent sub-questions when necessary. This ensures that complex queries are manageable and aligned with the KG structure. The second prompt matches entities identified in each sub-question with corresponding classes from the KG, ensuring that relevant components and, consequently their relationships and properties, are identified for further processing. The final prompt

```
PREFIX beo:<https://pi.pauwel.be/voc/buildingelement#>
PREFIX props: <http://lbd.arch.rwth-aachen.de/props#>

SELECT DISTINCT ?batid ?length ?height ?area
WHERE {
  ?wall a beo:Wall .
  ?wall props:batid_attribute_simple ?batid .
  ?wall props:length_property_simple ?length .
  ?wall props:unconnectedHeight_property_simple ?height .
  ?wall props:area_property_simple ?area .
}
```

Fig. 5. Generated SPARQL query for the first sub-question of the example.

generates SPARQL queries based on the identified classes matched with each sub-question, constructing queries that efficiently and accurately retrieve the required data for each sub-question. Together, these prompts ensure a systematic and precise transformation of user questions into actionable queries. Below, each of the three prompts is explained in detail.

The first prompt involves simplifying the user question and, if it is considered complex, breaking it down into sub-questions. Complex questions involve multiple entities, or interconnected aspects, often requiring division into sub-questions to facilitate comprehensive and accurate answers across different dimensions or categories. Simple questions focus on a single entity, or aspect, requiring a straightforward response from a specific category of information. For example, a question like *How many stairs are there on the second floor?* is considered a simple question since it requires information only from the `beo:Stair` entity. On the other hand, a complex question such as

What is the biggest size window that could fit in the smallest wall?

involves comparing multiple entities (`beo:Window` and `beo:Wall`) as well as attributes, such as `props:area_property_simple` for both and `props:overallHeightIfcWindow_attribute_simple` `props:overallWidthIfcWindow_attribute_simple` for `beo:Window` and `props:unconnectedHeight_property_simple` for `beo:Wall`. This question would be therefore split into the two following sub-questions using the Simplify Question Prompt shown in Fig. A.10, which employs a 3-shot learning strategy:

- *What is the size of the smallest wall?*
- *What are the maximum sizes of windows that can be fitted in a wall?*

Once the sub-questions are generated, the next step is to identify the relevant classes, relationships, and properties previously extracted from the KG that can be used to respond to each of them. We use the Identify Relevant Classes Prompt illustrated in Fig. A.11 to extract the most relevant KG classes that match entities within the user's sub-question. In the prompt, we would replace the "sub question" variable with the text corresponding to the underlying sub-question and the "classes" variable with the classes previously extracted at the end of the first block of the pipeline. Applied to the two sub-questions of the example above, this prompt would identify the `beo:Wall` class for the first sub-question and `beo:Window` class for the second sub-question.

Once the relevant classes are identified, we extract all their associated relationships and properties from the KG. Combined with the related sub-question, this set of triples serves as context for the next step, where the LLM generates a SPARQL query for each sub-question to retrieve the necessary data to answer the sub-question using the Construct SPARQL Query Prompt shown in Fig. A.12. Specifically, following the example used throughout this section, the prompt takes the class `beo:Wall` for the first sub-question and `beo:Window` for the second sub-question and their associated relationships and properties, the text of the sub-questions, and a list of `prefixes` with their definitions, extracted directly from the Turtle file header of the building KG. These inputs ensure the LLM is equipped with precise contextual information to construct a syntactically and semantically accurate SPARQL query for each sub-question.

In line with our example, the generated SPARQL queries to retrieve the data necessary for determining the size of the smallest wall (first sub-question) and the largest dimension of a window that can fit within a wall (second sub-question) are presented in Figs. 5 and 6.

In addition to these examples, the system is also capable of producing more complex SPARQL queries that exploit the graph nature of the data. For instance, in response to the question *"Are there any rooms connected to more than one room through doors?"*, the LLM generates a query that navigates multiple graph hops, identifying all rooms that share door elements with more than one distinct neighboring room. This type of query goes beyond simple attribute filtering and demonstrates the ability of the system to discover relational patterns and structural dependencies across interconnected entities within the KG.

The query shown in Fig. 7 highlights how ASK-BIM leverages the graph structure of the building knowledge base to perform multi-entity reasoning, traversing relationships among rooms and shared elements to derive higher-level semantic insights.

In the next block, the generated SPARQL queries will be executed to obtain a broad set of results that can be further interpreted and processed to identify the specific information needed to answer the complete user's query.

```
PREFIX beo: <https://pi.pauwel.be/voc/buildingelement#>
PREFIX props: <http://lbd.arch.rwth-aachen.de/props#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bot: <https://w3id.org/bot#>

SELECT ?windowSizeLabel ?overallHeight ?overallWidth
WHERE {
 ?window rdf:type beo:Window .
 ?window props:overallHeightIfcWindow_attribute_simple ?overallHeight .
 ?window props:overallWidthIfcWindow_attribute_simple ?overallWidth .
 ?window rdfs:label ?windowSizeLabel .
}
```

Fig. 6. Generated SPARQL query for the second sub-question of the example.

```
SELECT ?room (COUNT(DISTINCT ?otherRoom) AS ?connectedRoomCount)
WHERE {
  ?room rdf:type beo:Room .
  ?room props:containsElement ?door .
  ?door rdf:type beo:Door .
  ?otherRoom rdf:type beo:Room ;
             props:containsElement ?door .
  FILTER(?otherRoom != ?room)
}
GROUP BY ?room
HAVING (COUNT(DISTINCT ?otherRoom) > 1)
```

Fig. 7. Example of a multi-hop SPARQL query generated by the LLM to identify rooms connected to more than one other room through doors.

4.3. Aggregation and final answer generation

The third block of our pipeline involves executing the generated SPARQL queries, output of the second block, on the SPARQL endpoint, retrieving the results for each sub-question, and aggregating them. The data thus retrieved are then provided as context to a final LLM prompt, which combines the data with the original user question to produce a comprehensive and accurate natural language response. By mapping the classes of the KG to the entities of each sub-question and integrating the results with the reasoning capabilities of an LLM, this step ensures that complex queries are coherently resolved into a unified answer aligned with the user's intent.

For our example, a portion of the results from the two sub-questions is shown in Figs. A.14 and A.15, respectively. The results are returned in JSON format by GraphDB, the SPARQL endpoint used.

The query results of each sub-question are then combined and provided as context to the Final Response Generation Prompt shown in Fig. A.13, which interprets the data and answers the original user question. For our example, the LLM produces the response illustrated in Fig. 8.

This process ensures that the answers are complete and accurate and allows the system to handle nuanced queries involving calculations, comparisons, or logical reasoning.

5. Methodology to define questions

To define the questions for evaluation within our pipeline, we engaged in discussions with architects and civil engineers involved in the context of CHRONICLE and LEGOFIT, two European projects that have shaped this research, as outlined in Section 1. Additionally, we consulted BIM modelers from both projects who participated in the Scan2BIM process. Our objective was to compile an initial set of questions that are both relevant and meaningful to professionals in the field. The questions represent information that a user of a Common Data Environment (CDE) or DBL might want to obtain.

```
<data_interpretation>
User's question: What is the biggest size window that I could
fit in the smallest wall?

Key data points from the SPARQL results:
1. Wall area: 0.21822289677445708 square meters - This indicates
the size of the smallest wall available for installation.
2. Window sizes available (height x width):
- 1200 x 2200 mm (2.64 m^2)
- 1200 x 2100 mm (2.52 m^2)
- 2100 x 3200 mm (6.72 m^2)
- 1200 x 3200 mm (3.84 m^2)
- Other window sizes, all greater than the area of the wall.

Calculations: The smallest wall area of ~0.218 m^2 makes it
clear that no standard window sizes listed, which are all larger
than the wall area (above 2.5 m^2), can fit.

Data assessment: The data clearly outlines various window sizes,
but none can fit within the small wall's area due to their
significantly larger dimensions.

Potential limitations: The query results contain overlapping
window data, but the fundamental issue is that all window sizes
exceed the wall area, making it impossible to fit any of them.
</data_interpretation>

<answer>
Unfortunately, no window sizes can fit in the smallest wall, as
the smallest wall area is approximately 0.218 m^2, while all
listed window sizes are significantly larger than this area.
</answer>
```

Fig. 8. Final response to the user's question of example.

Specifically, the objective was to formulate questions that required retrieving information directly from the IFC model, with some of them also needing additional inference or reasoning to generate a complete answer.

For example, consider the question: *Analyze and justify if the number of windows in this house is reasonable*. Answering this requires extracting the number of window elements from the IFC model and assessing their distribution within the building. However, determining whether this number is reasonable involves an additional reasoning step, where external contextual knowledge, such as architectural norms or typical building design standards, must be considered.

Another illustrative example is: *What is the highest riser on the stairs of the building, and is it reasonable or too high?* In this case, the IFC model provides data regarding the riser heights in the building, but evaluating whether the highest riser is within acceptable limits requires inference based on construction regulations or ergonomic considerations. As in the previous example, the LLM must supplement the IFC-extracted data with external knowledge to determine compliance or usability.

This kind of mixed-question format allows for flexibility and universality across different building models and topologies. It is important to note that not all questions need to have both a data extraction component and an inference component; some can be answered solely based on the IFC model, while others may complement previous information without requiring additional data from the model. We also used variations of the questions by seeking synonyms for keywords or using LLMs to reformulate and rephrase them.

As previously mentioned, our objective was to demonstrate the concept in open BIM models with geometric data and material information. An example can be seen in the IFC excerpt included in Fig. A.16, where metallic surfaces (#4674) and wooden surfaces (#4681) are displayed. The approach of integrating reliable information within the IFC model complemented by LLM capabilities, is scalable and compatible with similar questions from models incorporating other dimensions as well.

Finally, we selected a mix of simple and complex questions. Some focused on single IFC elements, while others required linking two or more elements. We also made sure to include questions that used only BIM data and others that needed extra information from the LLM. Since the validated IFC lacked data from certain dimensions, we excluded questions related to cost analysis and safety. In the end, we came up with a set of 28 questions with different levels of complexity for our experiments.

6. Experimental settings

In this section, we outline the settings used in our performance evaluation. To evaluate the tool's effectiveness, we categorized the questions based on their complexity. Complexity was determined by factors such as the number of distinct entities matched in the KG (e.g., windows, doors), the relationships between these entities, their properties, and whether the question required inference from the LLM to be answered. Given these factors, the complexity of a question increases when:

- A greater number of distinct entities of the KG must be recognized and queried.
- More relationships between these entities must be identified and interpreted.
- A wider range of properties, either for single entities or across relationships, must be retrieved and reasoned upon to respond to the user's question.

These entities, the relationships between them, and their properties may not always be explicitly mentioned in the user's question but might be inferred and recognized by the LLM to construct the correct SPARQL query.

The validation process was designed to generate intermediate results at each stage of the pipeline, enabling a comprehensive evaluation of its performance. It can be divided into three key steps:

- **Entity-Focused Sub-question Generation**: The system's capability to decompose the original question into distinct entity-focused sub-questions was evaluated. This step ensures that each entity referenced in the original query is addressed through a dedicated sub-question, with a focus on that entity and its associated properties and relationships. The evaluation assesses how effectively the system identifies and distinguishes different entities in the question, ensuring that it generates only the necessary sub-questions to retrieve the required data for answering the original question.
  Each entity-focused sub-question must incorporate the necessary entities to address a specific part of the original query. Importantly, the complete set of sub-questions must collectively represent the full scope of the original query, ensuring no critical aspects are omitted. Achieving this balance is critical, as questions that are overly fragmented can dilute the relevance and clarity of the subsequent steps, while insufficiently divided questions risk omitting necessary details and connections. This step aims to ensure the coherence and effectiveness of downstream processes in the pipeline.
- **SPARQL Query Generation**: The generation of SPARQL queries for each sub-question was assessed. The evaluation focused on whether the generated SPARQL queries were syntactically correct, logically sound, and capable of retrieving the necessary data for each sub-question. SPARQL queries that failed due to syntax errors or missing logical connections were classified as errors.
- **Reasoning on Retrieved Data**: The final step involved passing the results of the SPARQL queries of each sub-question, along with the original question, to the LLM for reasoning and generating the final answer. This step was evaluated based on how effectively the system reasoned about the retrieved data to produce a coherent, accurate, meaningful, and user-friendly response.

7. Question classification

We classified user questions along two orthogonal dimensions that capture both the complexity of the questions in natural language and how directly the required information is represented in the KG.

1. **Entity and Property Involvement:**
   This dimension addresses the complexity of a question based on how many entities are involved and the complexity of the relationships among them. We define three levels:

   - **Single Entity, Single Property**: These questions include a single property of one entity. No additional inference or multi-entity reasoning is involved. An example is:

     "How many doors are on the second floor?"

     Here, the text *doors* corresponds to the `beo:Door` entity of the KG, and the text *second floor* to the `props:level_property_simple` property.
   - **Single Entity, Multiple Properties**: These questions focus on a single entity but require retrieving or comparing several of its properties. An example is:

     "What is the total surface area of all walls on the second floor?"

     In this case, the text *walls* corresponds to the `beo:Wall` entity of the KG, while the text *surface area* and *second floor* correspond to `props:area_property_simple` and `props:level_property_simple` properties in the KG.

- **Based on Reasoning**: These questions involve one or multiple entities, properties of them that might be present (explicitly or implicitly), and one or more relationships between the entities that require multi-layered reasoning, inference, and data aggregation to derive the correct answer. An example is:

  "What is the window-to-wall ratio in the building?"

  Here, the text *window-to-wall* matches with the `beo:Window` and `beo:Wall` entities of the KG, and their explicit property `props:area_property_simple`. ASK-BIM must determine the correct relationship between the entities to accurately answer the question, which, in this case, involves calculating the ratio by dividing the total window area by the total wall area.
  Another example is:

  "Are there any irregular-shaped rooms?"

  In this case, although the question explicitly refers to *rooms*, the entity room is not directly represented in the KG. Instead, rooms must be inferred from the relationships between multiple entities represented in the KG, such as `beo:Wall`, `beo:Slab`, and `ifc:IfcOpeningElement`. This requires reasoning beyond directly linked properties, integrating geometric and spatial relationships to accurately infer room boundaries and determine irregularity.

2. **Entity Representation in the Knowledge Graph:**
   This dimension assesses whether the questions target concepts and relationships that are explicitly encoded in the KG or require inference on other concepts.

- **Direct Questions**: These rely on entities or properties that are explicitly modeled as classes or direct attributes in the KG. Answering them requires only retrieving or comparing information already encoded in the KG, without significant inference about non-existent or implicit classes or relationships. For example:

  "What is the total number of treads in the building?"

  Here, the text *treads* is logically linked to the `beo:Stair` entity in the KG and the text *total number* to the `props:numberOfTreads_property_simple` property and requires counting how many are present for each `beo:Stair` entity. Since `props:numberOfTreads_property_simple` is a property explicitly represented in the KG, this question is considered direct.
- **Indirect Questions**: In contrast, indirect questions require inferences about concepts not directly represented as classes, relationships, or properties in the KG. For example:

  "Which corridors have doors wider than 900 mm?"

  Here, the text *doors* corresponds to the entity `beo:Door`. While `beo:Door` is a class explicitly defined in the KG, the text *corridors* does not have a corresponding entity in the KG. Therefore, this question is considered indirect and requires complex inference.

In ontology engineering, Competency Questions (CQs) are used to define, validate, and evaluate an ontology's capability to answer domain-relevant questions [26]. Following the taxonomy proposed in [26], we can map our evaluation questions to five CQ categories:

- Scoping CQs (SCQ) – Questions that delimit the coverage or boundaries of the BIM ontology, such as "Which building elements are represented in the model?"
- Validating CQs (VCQ) – Questions that test whether the ontology and KG contain sufficient information to support domain-relevant tasks, e.g., "How many doors are on the second floor?"
- Relationship CQs (RCQ) – Questions concerning interactions among entities, e.g., "Which walls have openings, and what are their sizes?"
- Foundational CQs (FCQ) – Questions linked to high-level alignment with domain abstractions (e.g., "What defines a structural element in the model?"), which are less frequent in our dataset.
- Metaproperty CQs (MpCQ) – Questions that involve reasoning about properties of classes or constraints, such as "Are there any irregular-shaped rooms?"

Overall, the majority of our 28 questions fall under the Validating and Relationship categories, reflecting ASK-BIM's dual purpose of (i) assessing the ontology's ability to represent BIM entities and their interrelations and (ii) enabling natural-language access to such information. This mapping clarifies the role of our question set in evaluating the ontology-driven artifact and aligns our approach with established ontology-engineering methodologies.

7.1. Results and observations

The evaluation results, summarized in Fig. 9, provide an overview of the system's performance across the 28 questions defined in Section 5 according to the two classifications just illustrated.

Three expert annotators specializing in the BIM domain evaluated each response, classifying them as either correct (i.e., the answer provided by ASK-BIM is complete and responds entirely to the user's question) or incorrect (i.e., the answer is either partial

Single Entity, Single Property	Single Entity, Multiple Properties	Based on Reasoning	
How many doors are there on the second floor of the building? Direct	What is the total surface area of all the walls on the second floor? Direct	Which windows are installed on walls with a height greater than 3 meters? Direct	Which walls in the building are fire-resistant? Indirect
What is the size of the smallest window in the building? Direct	What is the highest riser on the stairs of the building? Direct	What is the biggest size window that I could fit in the smallest wall? Direct	The number of windows in the building can be considered reasonable? Indirect
What is the total number of treads in the building? Direct	Can you generate a detailed description of a window in the building using all its available properties and values? Direct	What is the total structural load supported by load-bearing walls on each floor? Indirect	Which walls are supporting other structural elements in the building? Direct
What is the height of the tallest column in the building? Direct	Which properties of windows are related to their orientation or position in the building? Direct	What is the total area enclosed by the walls on the second floor? Indirect	What is the window-to-wall ratio in the building? Direct
How many rooms are located on the second floor? Indirect	What is the fire rating of the largest door in the building? Direct	Are there any rooms connected to more than one room through doors? Direct	What is the ratio of usable floor area to the gross area for the second floor? Indirect
Which storey has the highest number of doors? Direct	Which walls on the ground floor are classified as load-bearing and have a height greater than 3 meters? Direct	Which ramps in the building are accessible according to universal design standards? Indirect	Are there any irregular-shaped rooms? Indirect
What is the average ceiling height of all hallways in the building? Indirect	What is the total glazed area of windows on the second floor? Indirect	Which corridors have doors wider than 900mm? Indirect	How many windows are in each of the facades? Indirect

Fig. 9. Classification of ASK-BIM responses based on correctness and complexity. Green cells represent fully correct answers (all reasoning steps executed correctly), yellow cells indicate partially correct answers (the SPARQL query generation step omitted some results), and red cells denote incorrect answers (failure across all steps). The classification also distinguishes between direct and indirect questions, highlighting the system performance in handling explicit versus inferred properties and relationships within the KG. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

or wrong). Their assessments were in complete agreement, achieving a 100% consensus. Correct responses were labeled as green, while incorrect ones were further categorized into yellow and red by us. Specifically:

- **Green**: The answer to the user's query is correct, indicating that all three steps outlined in Section 6 were performed accurately.
- **Yellow**: The answer is partially correct. While the three steps outlined in Section 6 were executed and a response was generated, the SPARQL query generation step omitted certain elements, which were nonetheless reflected in the response.
- **Red**: The answer is either empty or misses important elements. This usually arises from issues such as SPARQL queries failing to execute or retrieve relevant data, or the reasoning step being unable to generate a coherent response.

Out of the 28 questions, 16 were classified as green, 3 as yellow, and 9 as red. In the following subsections, we analyze each of these categories in greater detail, showing some examples for each category to illustrate the strengths and limitations of ASK-BIM.

7.1.1. Single entity, single property results

This category represents the most straightforward type of questions tested in our system. The simplicity of these queries is reflected in the obtained results. Among the seven questions in this category, five received responses classified as green, one as yellow, and one as red.

The yellow-classified case corresponds to the question *What is the average ceiling height of all hallways in the building?* In this instance, the pipeline failed to correctly construct the SPARQL query to successfully retrieve the heights of walls (whose corresponding entities in the KG are the class `beo:Wall` and the property `props:unconnectedHeight_property_simple`) specifically related to hallways (not corresponding to any entities in the KG) and instead retrieved the heights of all walls within the building. As a result, the system generated an incomplete answer by computing the average height of all walls rather than distinguishing between those that formed hallways and those that did not.

For the red-classified case, *How many rooms are located on the second floor?*, the pipeline failed to generate a valid response. The SPARQL query attempted to combine multiple entities, such as `beo:Wall`, `bot:Storey`, and `beo:Door` to define the concept of rooms (which is not represented as an entity in the KG), but did so incorrectly, resulting in an overly complex query that retrieved no results. This failure highlights the system's difficulty in correctly structuring queries when the target entity, in this case, **rooms**, is not explicitly represented in the KG and must be inferred through indirect relationships and properties.

Both questions posed significant challenges due to their indirect nature, requiring reasoning beyond explicitly defined entities in the KG.

7.1.2. Single entity, multiple properties results

This second category, while increasing in complexity by requiring multiple properties, remains relatively straightforward due to the involvement of only a single entity. This is reflected in the results: out of seven questions, six were classified as green, and one was classified as yellow.

The yellow case, *'What is the total glazed area of windows on the second floor?*, highlighted both the system's strengths and limitations. While the system correctly identified the `beo:Window` entity, the SPARQL query generation failed to retrieve the specific glazed area and instead returned the total `props:area_property_simple` property, which includes both glazed and non-glazed portions, leading to an incomplete answer.

This limitation arises because the glazed area is not explicitly stored as a distinct property in the KG, making the query inherently indirect. Instead, determining the glazed area of a window requires complex reasoning that involves calculating specific coordinate values of other elements, which are represented in the KG but not readily interpretable by an LLM. As a result, the system defaulted to retrieving the closest available property, `props:area_property_simple`, highlighting its difficulty in performing advanced reasoning beyond explicitly encoded attributes when key properties must be inferred.

7.1.3. Based on reasoning results

This category represents the most challenging, where responding correctly requires a significant level of reasoning, inference, and often multi-layered analysis across different entities in the KG. Due to the wide variety of possible reasoning-based queries, we included a total of 14 questions in this category. The complexity of these questions is reflected in the results: among the 14 tested, 5 were classified as green, 1 as yellow, and 8 as red.

The green cases, although requiring complex reasoning, were successfully answered by the system due to their direct nature. For example, in the question *What is the biggest size window that I could fit in the smallest wall?*, ASK-BIM correctly identified the two entities involved, `beo:Wall` and `beo:Window`. Then, the pipeline was able to correctly break down the question into two sub-questions and retrieve the dimensions of all windows and the walls in the building (`props:area_property_simple` for both and `props:overallHeightIfcWindow_attribute_simple` and `props:overallWidthIfcWindow_attribute_simple` for `beo:Window` and `props:unconnectedHeight_property_simple` for `beo:Wall`). The system then generated two correct SPARQL queries to extract this information and, in the final step, reasoned over the retrieved values to determine the largest window that could fit within the smallest wall. The success in such cases highlights the system's ability to handle complex queries effectively when all required properties are explicitly encoded in the KG.

One particularly notable case is the question *Which walls in the building are fire-resistant?*, which is the only indirect question classified as green. The system successfully inferred the fire resistance of walls, despite this property not being explicitly stored in

the KG. Instead, it extracted this information from the wall labels, which contain implicit indications of fire resistance. For instance, a triple indicating a sample wall label retrieved from the KG is:

`inst:wall_cf05557fa00240929c7c2b65baed39ad rdfs:label "BasicWall Interior 79 mm Partition (1-hr) 470096".`

The system first generated a correct SPARQL query to retrieve the wall labels. In the reasoning step, it successfully interpreted the notation `"(1-hr)"` within the label, correctly inferring that this indicates a one-hour fire rating, meaning the wall can withstand fire for one hour according to standard fire resistance tests. This example highlights the system's ability to extract relevant information from values within the KG and apply reasoning to derive implicit properties.

Moving to the red cases, the majority of failures occurred in the SPARQL query generation step. Due to the complexity of the questions, the system either generated an overly complex query that yielded no results or produced a generic query that retrieved illogical or irrelevant data, making it impossible to construct a coherent answer. An example of this issue is in the question *What is the total area enclosed by the walls on the second floor?*. In this case, the system generated a SPARQL query that retrieved the `props:area_property_simple` property values of individual walls on the second floor, rather than attempting to infer the enclosed area between them. As a result, this data was sent as context to the reasoning step, leading to an incorrect response.

These findings indicate that while ASK-BIM performs well in structured reasoning tasks, further improvements are needed to enhance its ability to construct meaningful SPARQL queries for more complex queries requiring implicit inferences or certain types of reasoning.

8. Conclusions and future work

Building Information Models are transforming the construction industry. These models provide a framework for visualization, collaboration, and information exchange among stakeholders involved in stages ranging from building design to demolition. BIM began as a way to exchange basic building data along with geometric information, but currently, they are widely used to exchange multidimensional information, including sustainability information, project planning data, cost analysis, or details about safety risks. This is leading to, on one hand, an immense source of information about buildings, and on the other, having BIM files growing beyond tens of megabytes to reach the hundreds. With this level of complexity and size, it becomes more difficult to consume the information within them and quickly retrieve specific data.

This prompted us to explore whether, given the advancements in AI and NLP, information could be extracted from these models more quickly and intuitively. Our goal is to combine the reliability of structured data with the extensive knowledge of LLMs, enabling stakeholders with diverse roles and expertise to access information more efficiently through a conversational interface and generate results tailored to their specific needs.

Current LLMs and modern chatbots are not far from being able to solve this problem, but there are still hallucinations and challenges when processing large files, where information may be scattered across different parts of the document storing the model. To address this, we have designed ASK-BIM, a pipeline based on semantic techniques, Chain of Thought inspired reasoning strategies and LLMs to transform BIM models into KGs and answer natural language queries. This facilitates the extraction of information, enriches it with relevant data, and even reasons about it using the latest LLMs.

Specifically, the pipeline starts with a conversion step from BIM IFC to a KG. It then uses a query analysis module to detect relevant properties, entities, and relationships and maps them to the KG. In this step, prompt generation techniques are used to create SPARQL queries that retrieve the necessary data. Finally, the retrieved information is aggregated, and the LLM is leveraged again to generate the final response.

We have applied this approach to a collection of BIM models and dozens of questions, presenting in this work a specific case study of a BIM model related to a building in Barcelona. During the validation, we identified various complexities and categorized the questions based on the entities, properties, and relationships involved. Additionally, we classified them as direct or indirect questions, observing promising results, especially for direct questions.

We observed that many partial or incorrect answers were due to incorrect retrieval of data from the KG with the SPARQL query generated. Additionally, in some cases, the indirect questions' results were not satisfactory especially when they required advanced spatial reasoning.

Such errors stem from a fundamental limitation: the tool lacks a complete and structured understanding of the underlying ontology. Due to prompt token constraints, it is not feasible to provide the entire ontology as context in the prompt sent to the LLM. This restriction limits the tool's ability to reason holistically about the available data, particularly when answering questions that require inferring relationships not explicitly represented in the KG.

Nevertheless, the satisfactory results for direct questions lead us to consider this approach as a viable method for combining existing information from the BIM model with external data from LLMs. This integration enables users to effectively query BIM models in natural language and receive responses that not only leverage embedded data from the BIM but also allow for enrichment with additional knowledge.

Taken together, the results discussed above provide clear evidence with respect to the research questions formulated in this work. In relation to **RQ1**, the transformation of BIM models into Linked Building Data KGs proves to be a key enabler for improving

semantic accessibility. By preserving explicit structural and spatial relationships among building elements, the KG representation allows BIM data to be queried in a more transparent and interpretable way, supporting natural language access without requiring detailed knowledge of IFC schemas.

With respect to **RQ2**, the ASK-BIM pipeline shows that LLMs can be effectively guided to translate natural language questions into structured graph queries when their reasoning is grounded in explicit semantic representations. The use of staged prompting strategies and schema-aware query generation enables controlled interaction with the KG, mitigating common issues such as hallucinations and misinterpretation that arise when LLMs are applied directly to raw BIM files.

Regarding **RQ3**, the experimental evaluation on real-world BIM models confirms that the integration of KGs and LLMs leads to improved correctness and expressiveness in query answering. The approach performs particularly well for direct questions involving explicit entities and properties encoded in the graph, while more complex and indirect queries, especially those requiring advanced spatial reasoning, remain challenging. In several cases, inaccuracies in SPARQL query generation resulted in partial or incomplete data retrieval, highlighting current limitations of the approach.

These limitations are largely attributable to the inability to provide the full ontology as contextual input to the LLM due to prompt size constraints, which restricts holistic reasoning over all available concepts and relationships. Despite this, the overall results indicate that ASK-BIM represents a viable and effective method for combining structured BIM data with the reasoning capabilities of modern language models. By grounding LLM responses in explicit semantic representations, the system enables interpretable and extensible natural language interaction with BIM models, while also supporting the enrichment of model data with external knowledge.

Further progress can be achieved by strengthening the interaction between the KG and the LLM, particularly in the extraction and aggregation of relevant graph data and in supporting more advanced spatial reasoning. Exploring agent-based reasoning strategies, improving SPARQL query generation, and evaluating local or open-source language models represent promising directions to enhance robustness, scalability, and cost-effectiveness.

Future work will focus on enhancing the extraction of relevant data from the KG, leveraging agentic AI solutions to improve reasoning over the data, particularly spatial data, and exploring methods to ensure more accurate and context-aware query responses.

Additionally, we plan to benchmark local and open-source language models as cost-effective alternatives to commercial APIs, and to explore optimizations in SPARQL query generation and its integration with LLMs to enhance reasoning, inference, and decision-making capabilities.

CRediT authorship contribution statement

Andrea Ibba: Writing – original draft, Validation, Software, Investigation. **Rubén Alonso:** Writing – review & editing, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization. **Diego Reforgiato Recupero:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research has been partially funded by the European Union's Horizon research and innovation programme, as part of CHRONICLE project - Building Performance Digitalisation and Dynamic Logbooks for Future Value-Driven Services (Grant agreement ID 101069722). This work has been partially funded by the European Union's Horizon project LEGOFIT - Adaptable technological solutions based on early design actions for the construction and renovation of Energy Positive Homes (Grant agreement ID: 101104058).

Appendix A. Prompts

The following prompts are structured according to the domain-specific prompting taxonomy developed for ASK-BIM, which builds on the TELeR taxonomy [25]. Each prompt supports a specific functional stage of the pipeline — decomposition, mapping, translation, or reasoning — corresponding respectively to the TELeR dimensions of task, representation, and language reasoning. This systematic organization ensures consistency, modularity, and theoretical grounding of the prompting design.

```
Your task is to divide the user's question into simpler,
independent sub-questions if necessary.

User Question: {user_question}

Instructions:
1) Simplify the Question if Needed:
- If the original question is complex and involves
multiple entities or relationships, break it into simpler,
independent sub-questions.
- Each sub-question should be simple enough that the data
required to answer it can be retrieved using the most
straightforward SPARQL query.
2) Handle Simple Questions:
- If the question is already simple and involves only one
entity or aspect, return it as is.
3) Avoid Overlap Between Sub-Questions:
- Sub-questions must not reference or rely on the results
of other sub-questions. Each sub-question must be
self-contained and independent.
4) Separate Entities per Sub-Question:
- Each sub-question should focus on a single entity or
aspect of the original question.
- For example, instead of combining entities or
relationships in a single sub-question, divide them to
focus individually on each.
5) Single Building Context:
- Assume that the question refers to a single building
unless explicitly stated otherwise.
6) Don't create sub-questions that involve reasoning:
- The objective of each sub-question is to retrieve data
from the KG, which will later be used in other interactions
for reasoning to ultimately answer the original question.
- Each sub-question should focus on retrieving only general
data about an entity that can help answer the question.
- Other sub-questions will retrieve additional parts of the
required data.

Output Format:
- If the question is simple:
Question 1: Return the original question
- Otherwise: Multiple questions formatted as follows:
Question 1: [Simplified Question 1]
Question 2: [Simplified Question 2]

Examples:
Original Question: What are the heights of all doors and
the total area of all walls?
Question 1: What are the heights of all doors?
Question 2: What is the total area of all walls?
Original Question: Which floors have more than five rooms,
and how many windows are there on those floors?
Question 1: Which floors have more than five rooms?
Question 2: How many windows are there on each floor?
Original Question: Which walls have openings, and what is
the size of the largest opening?
Question 1: Which walls have openings?
Question 2: List the size of the openings in walls
```

Fig. A.10. Simplify Question Prompt. It corresponds to the Decomposition category (task/language reasoning).

```
You are an AI assistant specialized in analyzing
building-related queries and identifying relevant classes
from a building KG. Your task is to determine which classes
are of high relevance to retrieve data for answering a
user's question.

First, review the following information:
1.  List of available building element classes:
<classes_list>
{classes}
</classes_list>

2.  User's question:
<user_question>
{sub_question}
</user_question>

Your goal is to identify the essential classes from the
provided list that are necessary to retrieve the data
needed to answer the user's question.

Please follow these steps:
1.  Analyze the user's question and the list of classes.
2.  Identify classes that can contain the necessary data to
answer the user question.
3.  Infer any additional classes that might be necessary to
provide a complete answer, even if they're not explicitly
mentioned.
4.  Compile a final list of relevant classes, including
only the most relevant.

Wrap your thought process inside <class_identification_process>
tags:
1.  Quote relevant parts of the user's question.
2.  List potentially relevant classes with a brief
explanation for each.
3.  Consider relationships between identified classes.
4.  Rank the relevance of each identified class (High,
Medium, Low).
5.  Summarize the final list of relevant classes and
explain your choices.

Instruction for your response output:
After completing the analysis, provide the final list of
relevant classes in a comma-separated format.
The output must be enclosed after the tag
<question_analysis>.
Ensure no additional text or explanation follows the
closing tag </question_analysis>.
Only include the class names in the comma-separated format
inside the closing tag, with no additional symbols or
formatting.

Output format example:
** Class1, Class2, Class3

Remember:
- Include only classes from the provided list.
- Take into account indirect relationships and inferred
data requirements, if needed, recognizing that classes
and properties within the knowledge graph may be
interconnected.
- In the final class list, include only classes with High
relevance.
- In the last line with the classes, always begin the line
with "**".

Now, please proceed with your analysis and provide the list
of relevant classes.
```

Fig. A.11. Identify Relevant Classes Prompt. It corresponds to the mapping category (representation/experience).

```
Your task is to create a valid SPARQL query based only on
the provided classes, properties, and prefixes to retrieve
the necessary data from the building knowledge graph to
answer the user's question.

Here is the structure of the knowledge graph:
<classes_and_properties>
{classes_and_properties}
</classes_and_properties>

Prefixes to be used:
<prefixes> {prefixes} </prefixes>

The user's question:
<question> {question} </question>

Instructions:
- Your primary goal is to create a simple and general
SPARQL query that retrieves a comprehensive set of data
related to the question.
- Avoid directly answering the question in the query.
Instead, retrieve a relevant set of values that can be
interpreted later to provide the answer.

Before constructing the final query, think through the
process carefully inside <query_planning> tags:
- Key Concepts and Relationships:  Identify the essential
entities and relationships in the question.
- Relevant Classes and Properties:  List the provided
classes and properties that correspond to these concepts.

Query Structure:
- There is no fixed structure for constructing the query;
adapt the query based on the user's question and the
available data.
- Always include DISTINCT on props:batid_attribute_simple
when querying elements to ensure unique real-world objects
are returned and avoid duplications.
- Construct the query to interpret and represent abstract
concepts (e.g., relationships, dependencies, or inferred
data) as needed, ensuring that the data retrieved can
support answering the original question comprehensively.

Review:
- Ensure only the provided classes, properties, and
prefixes are used, and use as few of them as possible.
- You do not need to use all classes if they are not
necessary|prefer using a single provided class with its
attributes whenever possible.
- Verify that the query follows valid syntax for the SPARQL
endpoint.

Important Notes:
- Prioritize Simplicity:  Prefer simple queries that
retrieve broader sets of data over complex queries that
attempt to directly compute or filter the answer.
- Include Identifiers:  Always include the
props:batid_attribute_simple for every element in the query.
- Include Prefixes:  Always include the right prefixes from
the prefixes list.
- Avoid Over-Filtering:  Add constraints only when
essential to keep the query results relevant but
comprehensive.
- Always return a broad set of attributes in the query
that can provide sufficient context to reason and answer
the question, avoiding queries that return only a limited
number of attributes.

Finally, generate the SPARQL query within triple quotes,
following this format:
‘‘‘ Your Generated Query ‘‘‘
```

Fig. A.12. Construct SPARQL Query Prompt. It corresponds to the translation category (task/representation).

```
Your task is to answer user questions about buildings using
data retrieved from a knowledge graph via SPARQL queries.

Here is the user's question:
<user_question>
{user_question}
</user_question>

Below is the dictionary containing each query along with
its results:
<sparql_query_results>
{sparql_query_results}
</sparql_query_results>

Instructions:
- The sparql_query_results is a dictionary where each key
is a SPARQL query and the value is the corresponding result
from the knowledge graph.
- When analyzing the results, use both the queries and the
returned data to understand the elements being retrieved
and their relationships.

Simplified Steps:
1) Understand the User's Question and Query Results:
- Interpret the user's intent and the structure of the
returned data.
- Assume the results are relevant and comprehensive.
- If an expected attribute is missing, assume the list is
already filtered by that criterion or an equivalent.

2) Interpret and Define the Returned Elements:
- Identify each element type based on the IFC schema.
- Recognize associations such as spatial containment,
composition, connectivity, and property references (e.g.,
material assignments).
- Treat entries with the same props:batid_attribute_simple
as referring to the same real-world object to avoid
duplication.

3) Answer the Question by Reasoning or Displaying the Data:
- If the question requests a list or value, use the
provided data directly if already filtered and complete.
- Perform additional filtering or calculations only if
necessary to derive the final answer.

4) Check Completeness and Accuracy:
- Verify if the results cover all aspects required to
answer the question.
- If the data is incomplete or ambiguous, mention any
limitations and provide partial answers if possible.
```

Fig. A.13. Final Response Generation Prompt. It corresponds to the reasoning category (experience/language reasoning).

```
5) Formulate the Final Answer:
- Provide a concise answer that fully addresses the user's
question.
- Ensure the format matches the request (e.g., listing
elements if a list is requested).

Important Considerations:
- Do not mention the SPARQL queries or structure of the
data unless explicitly requested by the user.
- Use your IFC schema knowledge to interpret relationships
and attributes in the data.
- Assume that the queries provide relevant entities and
attributes needed to answer the question.

Output Format:
<data_interpretation>
User's question: Restate the user's question
Key data points from the SPARQL results:
1. Data point 1 - Definition: IFC schema definition -
Relevance to question: Explanation
2. Data point 2 - Definition: IFC schema definition -
Relevance to question: Explanation
...
Calculations: Any relevant calculations
Data assessment: Comments on completeness/reliability
Potential limitations: List any ambiguities or limitations
in the data
</data_interpretation>

<answer>
Concise answer to the user's question based on the analysis
</answer>

Example:
User's Question: "List all doors on the first floor."
SPARQL Results: A list of doors with their attributes
(including label and batId, but missing level).
Answer Logic:
- If the level attribute is missing, assume the list is
pre-filtered by the level and display all elements.
- If the level is present but unfiltered, apply filtering
in the reasoning step to list only first-floor doors.
- Always check for duplicate entries by comparing
props:batid_attribute_simple and consider them as the same
object if IDs match.
```

Fig. A.13. (*continued*).

```
{
    "batid" : {
      "type" : "literal",
      "value" : "361296"
    },
    "length" : {
      "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
      "type" : "literal",
      "value" : "12140.000001443696"
    },
    "height" : {
      "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
      "type" : "literal",
      "value" : "3460.0000000000905"
    },
    "area" : {
      "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
      "type" : "literal",
      "value" : "31.38064540692539"
    }
  },

  .
  .
  .

  {
    "batid" : {
      "type" : "literal",
      "value" : "349220"
    },
    "length" : {
      "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
      "type" : "literal",
      "value" : "14499.999999813655"
    },
    "height" : {
      "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
      "type" : "literal",
      "value" : "20540.00000000009"
    },
    "area" : {
      "datatype" : "http://www.w3.org/2001/XMLSchema#decimal",
      "type" : "literal",
      "value" : "231.70309999623487"
    }
}
```


Fig. A.14. SPARQL query results for the first sub-question.

```
{
    "windowSizeLabel" : {
        "type" : "literal",
        "value" : "Windows_Sgl_Plain:1200 x2200mm:462986"
    },
    "overallHeight" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#double",
        "type" : "literal",
        "value" : "2200."
    },
    "overallWidth" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#double",
        "type" : "literal",
        "value" : "1200."
    }
},

.
.
.

{
    "windowSizeLabel" : {
        "type" : "literal",
        "value" : "Windows_Sgl_Plain:1200 x2200mm:462986"
    },
    "overallHeight" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#double",
        "type" : "literal",
        "value" : "2200."
    },
    "overallWidth" : {
        "datatype" : "http://www.w3.org/2001/XMLSchema#double",
        "type" : "literal",
        "value" : "1200."
    }
}
```


Fig. A.15. SPARQL query results for the second sub-question.


```
#4669=IFCSTYLEDITEM(#4668,(#4659),$);
#4670=IFCCLOSEDSHELL((#4552,#4557,#4565,#4569,#4577,#4581,#4589,#4593,#46
#4671=IFCFACETEDBREP(#4670);
#4672=IFCCOLOURRGB($,0.40784313725490196,0.40784313725490196,0.4078431372
#4673=IFCSURFACESTYLERENDERING(#4672,0.,$,$,$,$,IFCNORMALISEDRATIOMEASURE
#4674=IFCSURFACESTYLE('Metal - Paint Finish - Grey',.BOTH.,(#4673));
#4675=IFCPRESENTATIONSTYLEASSIGNMENT((#4674));
#4676=IFCSTYLEDITEM(#4671,(#4675),$);
#4677=IFCCLOSEDSHELL((#4634,#4641,#4644,#4647,#4650,#4653));
#4678=IFCFACETEDBREP(#4677);
#4679=IFCCOLOURRGB($,0.65098039215686276,0.45490196078431372,0.2039215686
#4680=IFCSURFACESTYLERENDERING(#4679,0.,$,$,$,$,IFCNORMALISEDRATIOMEASURE
#4681=IFCSURFACESTYLE('Wood - Birch',.BOTH.,(#4680));
#4682=IFCPRESENTATIONSTYLEASSIGNMENT((#4681));
#4683=IFCSTYLEDITEM(#4678,(#4682),$);
#4684=IFCSHAPEREPRESENTATION(#101,'Body','Brep',(#4655,#4662,#4665,#4668,
#4685=IFCCARTESIANPOINT((50.,437.69999999999823));
#4686=IFCAXIS2PLACEMENT2D(#4685,#12);
#4687=IFCCIRCLE(#4686,800.);
#4688=IFCTRIMMEDCURVE(#4687,(IFCPARAMETERVALUE(180.00000000000017)),(IFCPA
#4689=IFCGEOMETRICSET((#4688));
#4690=IFCSHAPEREPRESENTATION(#105,'FootPrint','GeometricSet',(#4689));
#4691=IFCAXIS2PLACEMENT3D(#3,$,$);
#4692=IFCREPRESENTATIONMAP(#4691,#4684);
```


Fig. A.16. IFC model excerpt.

Data availability

Data will be made available on request.

References

[1] T. Yang, L. Liao, Research on building information model (BIM) technology, World Constr. 5 (2016) 1, http://dx.doi.org/10.18686/wcj.v5i1.1.

[2] G. Veerendra, S. Dey, E.J. Mantle, A.P. Manoj, S.S.A.B. Padavala, Building information modeling – simulation and analysis of a University Edifice and its environs – A sustainable design approach, Green Technol. Sustain. 3 (2) (2025) 100150, http://dx.doi.org/10.1016/j.grets.2024.100150, URL https://www.sciencedirect.com/science/article/pii/S2949736124000770.

[3] R. Charef, H. Alaka, S. Emmitt, Beyond the third dimension of BIM: A systematic review of literature and assessment of professional views, J. Build. Eng. 19 (2018) 242–257, http://dx.doi.org/10.1016/j.jobe.2018.04.028, URL https://www.sciencedirect.com/science/article/pii/S2352710217306320.

[4] A. Koutamanis, Dimensionality in BIM: Why BIM cannot have more than four dimensions? Autom. Constr. 114 (2020) 103153.

[5] R. Charef, H. Alaka, S. Emmitt, Beyond the third dimension of BIM: A systematic review of literature and assessment of professional views, J. Build. Eng. 19 (2018) 242–257.

[6] R. Kebede, A. Moscati, H. Tan, P. Johansson, Integration of manufacturers' product data in BIM platforms using semantic web technologies, Autom. Constr. 144 (2022) 104630, http://dx.doi.org/10.1016/j.autcon.2022.104630, URL https://www.sciencedirect.com/science/article/pii/S0926580522005003.

[7] Z. Wang, H. Ying, R. Sacks, Two fundamental questions concerning BIM data representation for machine learning, in: CIB W78 2024, 2024.

[8] N.H. Thuan, A. Drechsler, P. Antunes, Construction of design science research questions, Commun. Assoc. Inf. Syst. 44 (1) (2019) http://dx.doi.org/10.17705/1CAIS.04420.

[9] R. Alonso, R. Olivadese, A. Ibba, D.R. Recupero, Towards the definition of a European Digital Building Logbook: A survey, Heliyon (2023).

[10] T. Bloch, Connecting research on semantic enrichment of BIM - Review of approaches, methods and possible applications, J. Inf. Technol. Constr. 27 (2022) 416–440, http://dx.doi.org/10.36680/j.itcon.2022.020, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85133341225&doi=10.36680%2fj.itcon.2022.020&partnerID=40&md5=bdfc6f37b9e7773519e48cc003041c64.

[11] Y. Ding, J. Ma, X. Luo, Applications of natural language processing in construction, Autom. Constr. 136 (2022) 104169, http://dx.doi.org/10.1016/j.autcon.2022.104169, URL https://www.sciencedirect.com/science/article/pii/S0926580522000425.

[12] N. Wang, R. Issa, C. Anumba, NLP-based query-answering system for information extraction from building information models, J. Comput. Civ. Eng. 36 (2022) 04022004, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0001019.

[13] A. Nabavi, I. J. Ramaji, N. Sadeghi, A. Anderson, Leveraging natural language processing for automated information inquiry from building information models, J. Inf. Technol. Constr. 28 (2023) 266–285, http://dx.doi.org/10.36680/j.itcon.2023.013.

[14] J.-R. Lin, Z.-Z. Hu, J.-P. Zhang, F.-Q. Yu, A natural-language-based approach to intelligent data retrieval and representation for cloud BIM, Computer-Aided Civ. Infrastruct. Eng. 31 (1) (2016) 18–33, http://dx.doi.org/10.1111/mice.12151, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.12151, URL https://onlinelibrary.wiley.com/doi/abs/10.1111/mice.12151.

[15] M. Yin, L. Tang, C. Webster, S. Xu, X. Li, H. Ying, An ontology-aided, natural language-based approach for multi-constraint BIM model querying, J. Build. Eng. 76 (2023) 107066, http://dx.doi.org/10.1016/j.jobe.2023.107066, URL https://www.sciencedirect.com/science/article/pii/S2352710223012457.

[16] N. Wang, R.R.A. Issa, C.J. Anumba, Query answering system for building information modeling using BERT NN algorithm and NLG, in: Computing in Civil Engineering 2021, pp. 425–432, http://dx.doi.org/10.1061/9780784483893.053, arXiv:https://ascelibrary.org/doi/pdf/10.1061/9780784483893.053, URL https://ascelibrary.org/doi/abs/10.1061/9780784483893.053.

[17] S. Yang, M. Teng, X. Dong, F. Bo, LLM-Based SPARQL Generation with Selected Schema from Large Scale Knowledge Base, Springer, 2023, pp. 304–316, http://dx.doi.org/10.1007/978-981-99-7224-1_24.

[18] V. Emonet, J. Bolleman, S. Duvaud, T.M. de Farias, A.C. Sima, LLM-based SPARQL query generation from natural language over federated knowledge graphs, 2025, URL https://arxiv.org/abs/2410.06062.

[19] S. Tang, D.R. Shelden, C.M. Eastman, P. Pishdad-Bozorgi, X. Gao, A review of building information modeling (BIM) and the internet of things (IoT) devices integration: Present status and future trends, Autom. Constr. 101 (2019) 127–139.

[20] S. Jiang, L. Jiang, Y. Han, Z. Wu, N. Wang, OpenBIM: An enabling solution for information interoperability, Appl. Sci. 9 (24) (2019) 5358.

[21] I. buildingSMART, Specification Technology.

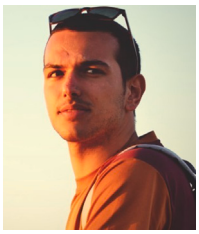
[22] M. Bonduel, J. Oraskari, P. Pauwels, M. Vergauwen, R. Klein, The IFC to linked building data converter-current status, 2018.

[23] M.H. Rasmussen, M. Lefrançois, G. Schneider, P. Pauwels, BOT: The building topology ontology of the W3C linked building data group, Semant. Web (2020) http://dx.doi.org/10.3233/SW-200385.

[24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q.V. Le, D. Zhou, et al., Chain-of-thought prompting elicits reasoning in large language models, Adv. Neural Inf. Process. Syst. 35 (2022) 24824–24837.

[25] S.K. Karmaker Santu, D. Feng, TELer: A general taxonomy of LLM prompts for benchmarking complex tasks, in: H. Bouamor, J. Pino, K. Bali (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, Singapore, 2023, pp. 14197–14203, http://dx.doi.org/10.18653/v1/2023.findings-emnlp.946, URL https://aclanthology.org/2023.findings-emnlp.946/.

[26] C.M. Keet, Z.C. Khan, Discerning and characterising types of competency questions for ontologies, 2024, URL https://arxiv.org/abs/2412.13688.



Andrea Ibba received the B.S. and M.S. degrees in computer science from the University of Cagliari, Italy, where he is currently pursuing the Ph.D. degree. His research interests include natural language processing, conversational agents, and Digital Building Loogbook technology. He is the Key Developer of AKS-BIM, a tool for answering natural language question about BIM models by leveraging LLM's capabilities and knowledge graph technologies.

Rubén Alonso received the Diploma degree in advanced studies on computing systems from the University of the Basque Country and the degree in computer science and engineering from the University of Deusto. He has been involved in industrial and research projects, since 2004, mainly related to computational trust, security, and embedded systems. He was part of the Trust and Security Unit, European Software Institute, and was responsible for the Security and Embedded Systems Group of Visual Tools. He was also a Visiting Researcher with the VTT Technical Research Centre of Finland and the University of Trento. Currently, he coordinates the ICT and Robotics Department of R2M Solution. His research interests include Linux embedded systems, robotics, IT security, and data mashups. He has experience in the European Research Programmes and industrial experience with various firms.

Diego Reforgiato Recupero received the Ph.D. degree in computer science from the University of Naples Federico II, Italy, in 2004. He has been a Full Professor with the Department of Mathematics and Computer Science, University of Cagliari, Italy, since February 2022. From 2005 to 2008, he was a Postdoctoral Researcher with the University of Maryland College Park, USA. He won different awards in his career (such as the Marie Curie International Reintegration Grant, Marie Curie Innovative Training Network, Best Researcher Award from the University of Catania, Computer World Horizon Award, Telecom Working Capital, Startup Weekend, and Best Paper Award). He co-founded seven companies within the ICT sector and is actively involved in European projects and research (with one of his companies he won more than 40 FP7 and H2020 projects). His current research interests include sentiment analysis, semantic web, natural language processing, human–robot interaction, financial technology, and smart grids. He is the author of more than 250 conference and journal articles in these research fields, with more than 4000 citations.