

Politecnico di Torino

Master's Degree in Building Engineering

A.y. 2025/2026

September 2026

BIM-LLM Integration for Facility Management: A Framework for Information Exchange in Maintenance Workflow

Supervisor:

Prof. Anna Osello

Candidate:

Milad Forghani

Co-Supervisor:

Dr. Nicola Rimella

Abstract

This thesis develops and evaluates a locally hosted framework that links Large Language Models (LLMs) with Building Information Modelling (BIM) for building maintenance. The study asks how an LLM-based interface can connect heterogeneous building-maintenance information to BIM while maintaining computational reliability and human control over model updates. The case study is the prefabricated ventilated facade of the Dumarey building at the Politecnico di Torino. Its information base combines a federated BIM model, regulatory and technical documents, and an inspection register during twelve inspection periods.

The framework combines retrieval-augmented generation (RAG) with task-specific tools and runs entirely on local infrastructure. Four locally executable LLMs were first compared with expert-defined reference answers. The selected prototype was then evaluated, with fidelity to the reference answer measured separately from consistency across repeated runs. The responses were therefore fairly stable, but stability did not always correspond to correctness. Independent execution traces were also examined to verify tool use and identify where failures occurred. Keeping the pipeline local prevents safety-related building-condition data from being sent to external inference services.

The final part of the workflow addresses the return of information to the BIM environment. Validated inspection findings are transferred through Dynamo to the instance parameters of the corresponding Revit elements, using the unique panel code as the shared asset identifier. The updated information is then exported back to the inspection register. Human confirmation is required before any change is committed, and IFC property sets are used to support interoperability. The results show that a two-way LLM-BIM maintenance workflow can be implemented under realistic institutional constraints. However, the prototype should remain a decision-support tool rather than an autonomous authority. The main contributions are the proposed workflow architecture, an evaluation approach that distinguishes accuracy from consistency, and a controlled method for keeping BIM condition information current.

Acknowledgements

I would like to express my sincere gratitude to Professor Anna Osello, my supervisor, for her guidance, support, and encouragement throughout this thesis. Her expertise and thoughtful feedback helped shape this work and gave me the opportunity to learn and grow throughout the research process.

I am especially grateful to Dr. Nicola Rimella, my co-supervisor, for his constant support and guidance throughout this journey. He was always available when I needed help, patiently answering my questions and helping me work through the challenges I encountered along the way. His advice, encouragement, and genuine dedication to this work have meant a great deal to me and played an important role in bringing this thesis to completion.

I would also like to thank my family and loved ones for their unconditional love, patience, and unwavering support. Being far from home has not always been easy, but their belief in me has always given me the strength to persevere and continue moving forward. I am deeply grateful for their constant encouragement and for being there for me throughout this journey.

As I come to the end of this journey, I carry with me a deep connection to Iran, my homeland, and the enduring hope of all those who long for a freer and brighter future. I carry the memory of a country rich in life, strength, and resilience, and I remember those who gave their lives in the pursuit of freedom, dignity, and happiness. May their voices never be forgotten and may the hope they carried for a better Iran live on.

Contents

Abstract 2
Acknowledgements 3
1. Introduction 10
2. Background and Literature Review 13
2.1 BIM as a Lifecycle Information System 13
2.2 BIM for Maintenance Planning, Facility Management, and Existing Buildings 15
2.3 From BIM Interoperability to Semantic Reasoning 17
2.4 LLMs, Agents, and RAG as Enabling Technologies 18
2.5 Existing BIM-LLM Integration in AECO Workflows 20
2.6 LLMs and BIM for Maintenance Workflow 22
2.7 Human Interaction and Maintenance Workflow Design 25
2.8 Evaluation, Validation, and Limitations 26
2.9 Research Gap and Positioning of the Thesis 27
3. Case Study: The Dumarey Building Facade 29
3.1 Why this building was chosen 29
3.2 Location and urban context 29
3.3 The building and its ventilated Balmoral facade 30
3.4 The maintenance problem 31
3.5 The BIM model as the geometric backbone 32
3.6 The Panel Inspection Register 32
3.6.1 Panel taxonomy and facades 32
3.6.2 Inspection history 33
3.6.3 Condition of the envelope 33
3.7 Supporting documentation 34
3.8 The panel code: the key that unifies the case study 34
3.9 Suitability of the case study 35
4. Methodology 36
4.1 Research design 36
4.2 Comparative selection of the local LLM 37
4.2.1 Objective and candidate models 37
4.2.2 Question set and reference answers 37
4.2.3 Scoring by embedding similarity 37
4.2.4 Results and model selection 38

4.3 Construction of the retrieval-augmented agent 40
4.3.1 Architectural principle: heterogeneous sources, heterogeneous handling 40
4.3.2 Prose ingestion and the vector store 41
4.3.3 The two tools 42
4.3.4 Agent orchestration and prompt engineering 43
4.3.5 Implementation stack 43
4.4 Evaluation of the RAG system 44
4.4.1 Evaluation question set 44
4.4.2 Repeated-run protocol 44
4.4.3 Metric 1 — Answer fidelity 45
4.4.4 Metric 2 — Self-consistency 47
4.4.5 Metric 3 — Manual four-class evaluation 48
4.5 Closing the loop: BIM integration and write-back 50
4.5.1 The BIM model as the authoritative condition record 51
4.5.2 The inspection parameter schema 52
4.5.3 The panel code as join key 53
4.5.4 Inspection capture and validation 53
4.5.5 The Dynamo graphs 54
4.5.6 Propagation to the documentation and the RAG corpus 58
4.5.7 Interoperability and governance 59
4.6 User interface: presenting the answer to the user 60
4.6.1 Role of the presentation layer 60
4.6.2 What must be displayed 60
4.6.3 Option A — Web application 61
4.6.4 Option B — Telegram bot 63
4.6.5 Comparison and selection 64
4.7 Methodological limitations 65
4.8 Summary 66
5. Results and Discussion 67
5.1 Selection of the local model 67
5.1.1 Why questions 9 and 14 are weak for every model 69
5.2 Answer fidelity 70
5.2.1 The strong end: definitional retrieval 72
5.2.2 The empty cell at question 4, run 7 72
5.2.3 The pale row at question 8 73

5.3 Self-consistency across runs 73
5.4 Manual four-class evaluation 75
5.4.1 Where correctness and cosine similarity disagree 77
5.5 Tool use and the composition limitation 78
5.6 Discussion 78
6. Conclusions and Future Work 80
6.1 Conclusions 80
6.2 Main Contributions 81
6.3 Limitations 82
6.4 Future Work 82
References 84
Appendix A: Model selection benchmark questions 86

List of Figures

Figure 2.1. BIM data extraction methodology for structuring asset information 14
Figure 2.2. Geometry and data combined through the IFC standard for exchange between BIM applications 15
Figure 2.3. Architecture of a Retrieval-Augmented Generation (RAG) system 19
Figure 2.4. Semantic interoperability framework for BIM-LLM integration..... 20
Figure 2.5. LLM-based framework for automated code-compliance checking in BIM 22
Figure 2.6. LLM- and BIM-driven framework for corrective maintenance task planning 23
Figure 2.7. LLM-based building O&M information-query workflow 24
Figure 2.8. The LLM agent as an interface between the user and the underlying system 26
Figure 2.9. RAG indexing and query workflow with seven potential failure points 27
Figure 3.1. The Dumarey building on Corso Castelfidardo, Turin, showing the Balmoral granite ventilated facade..... 30
Figure 3.2. Isometric view of the Dumarey BIM envelope model, with individual curtain-panel elements and facades A–D..... 31
Figure 4.1. Model selection benchmark: cosine similarity to expert answers across 20 questions and four candidate models (embeddinggemma:300m)..... 39
Figure 4.2. Hybrid RAG-agent architecture, combining semantic document retrieval with direct queries over the 5,496-panel inspection register. 41
Figure 4.3. Answer fidelity across 10 questions and 10 repeated runs, measured by cosine similarity to the expert key..... 46
Figure 4.4. Manual four-class evaluation of 100 answers: Correct (C), Wrong (W), Hallucination (H), and Error (E). 49
Figure 4.5. Two-way LLM-BIM maintenance workflow linking natural-language queries, retrieval, Dynamo integration, and controlled BIM write-back. 51
Figure 4.6. Dynamo integration layer for BIM-to-LLM data extraction and validated LLM-to-BIM write-back. 55
Figure 4.7. Dynamo export graph for BIM-to-Excel panel and inspection-data extraction. 57
Figure 4.8. Dynamo write-back graph for importing inspection data from Excel and updating the matching Revit panels. 57
Figure 4.9. Revit Panel Schedule after write-back, showing panel codes and the imported current inspection grade..... 58
Figure 4.10. Curtain Panel properties after write-back, showing the panel code and imported current-period inspection grade. 58
Figure 4.11. Conceptual chat-interface mock-up showing natural-language queries, cited answers, and tabular inspection-history output. 61
Figure 4.12. Interface mock-up, project view: the Dumarey building, project focus, and available data sources. 62

Figure 4.13. Interface mock-up, documents view: ingested regulations, reports, specifications, and inspection-register sources. 62
Figure 4.14. Interface mock-up, model view: a selected panel linked to its code, category, latest inspection, and current grade. 63
Figure 5.1. Model selection benchmark: cosine similarity to expert answers across 20 questions and four candidate models. 69
Figure 5.2. Answer fidelity across 100 responses (10 questions x 10 runs), measured against expert key answers. 71
Figure 5.3. Self-consistency across repeated runs, based on pairwise cosine similarity for each question. 74
Figure 5.4. Manual four-class evaluation of 100 answers, showing Correct, Wrong, Hallucination, and Error outcomes by question and run. 76

List of Tables

Table 3.1. Distribution of the 5,496 panels across the four facades 32
Table 3.2. Distribution of worst-observed defect grade (Max Grade Recorded) 33
Table 3.3. The case-study corpus and how each part is handled 34
Table 4.1. Structure of the research 36
Table 4.2. Candidate models 37
Table 4.3. Model selection benchmark results (cosine similarity to expert reference answers, 20 questions) 38
Table 4.4. Evaluation question set of the RAG system 44
Table 4.5. Mean answer fidelity per question across 10 runs 46
Table 4.6. Self-consistency across 10 runs (45 run pairs per question) 47
Table 4.7. Outcome of the manual four-class evaluation (100 answers) 49
Table 4.8. Inspection parameter schema (shared, instance-bound, Curtain Panels category) 52
Table 4.9. Validation rules applied to every inspection record 54
Table 4.10. Propagation of an update to the retrieval layer 59
Table 4.11. Comparison of the two interface options 64
Table 5.1. Model selection benchmark results (cosine similarity to expert reference answers, 20 questions) 67
Table 5.2. Mean answer fidelity per question across 10 runs, ordered from best to worst 71
Table 5.3. Self-consistency across 10 runs (45 run pairs per question), ordered from most to least consistent 74
Table 5.4. Outcome of the manual four-class evaluation (100 answers) 76
Table A.1. The BIM and facility-management questions used in the model selection benchmark 86

1. Introduction

Building Information Modelling (BIM) is now established in building design and construction. Its adoption has been encouraged by government mandates across Europe and by the development of a mature software market. Yet BIM was never intended to support design coordination alone. If model elements carry structured information such as materials, types, classification codes, installation dates, and maintenance requirements, the same model can also support operation and maintenance (O&M). In practice, this potential is only partly realised. Models are commonly prepared for design coordination and may not connect well with the asset-information systems used during operation. Maintenance evidence, meanwhile, is often spread across spreadsheets, document repositories, technical datasheets, and regulatory texts. The result is a familiar gap: the BIM model describes the building as designed, while the records needed to understand its present condition develop elsewhere [7,8].

For facility managers, the difficulty is therefore often less about missing information than about finding and combining what already exists. A manager may need to know how many facade panels have reached the replacement threshold, review the inspection history of one panel, or determine what a regulation requires after a particular defect grade is recorded. These questions can involve several sources, specialist BIM software, and manual interpretation. A useful maintenance interface must do more than search documents independently. It must connect an everyday question to the relevant BIM element, the correct structured records, and the technical evidence needed to interpret them.

Large Language Models (LLMs) offer a way to lower this interface barrier by allowing users to work in natural language. When they are coupled with Retrieval-Augmented Generation (RAG), they can base their responses on project-specific sources rather than only on information learned during training. Therefore, this can reduce hallucinations and make it possible to identify the source of an answer [20,25]. Recent studies have used LLMs to align natural-language questions with IFC classes and properties [4], generate structured BIM queries [5], and edit model information through tool calls [6]. Yet a fluent response is not necessarily a reliable maintenance answer. Interpreting a regulation or a material datasheet is different from filtering thousands of inspection records or calculating an aggregate. Reliability therefore depends on choosing the right information-processing method, executing it correctly, and leaving enough evidence for the result to be checked.

The return of maintenance information to BIM presents a second challenge. Most existing systems concentrate on retrieving information from a model or its associated documents, while inspection records continue to be updated separately. Unless new findings are brought back into the model, later queries

may rely on an outdated view of the asset. Closing this loop requires the system to receive inspection data, associate them with the correct element, validate their meaning, and update the authoritative record. This is also where control becomes critical. A wrong answer may mislead a user; a wrong written operation may silently change safety-relevant building data. Any update must therefore be traceable, validated, and approved by a person before it is committed.

These issues lead to the central research question of the thesis: How can an LLM-based interface connect heterogeneous building-maintenance information to BIM while maintaining computational reliability and human control over model updates? The question brings together three requirements: linking structured and unstructured project information through a shared asset identifier; producing answers with appropriate and verifiable computational operations; and returning validated findings to BIM without allowing the LLM to alter the model autonomously.

The thesis addresses this question by developing and evaluating a local, two-way LLM-BIM framework for maintenance. Facility managers can ask questions about project data in ordinary language and receive answers grounded in the available sources. Validated inspection findings can also be returned to the BIM environment. The framework is demonstrated on the prefabricated ventilated facade of the Dumarey building at the Politecnico di Torino. The case-study corpus includes a federated BIM model, an inspection register covering twelve inspection periods for 5,496 panels between 2017 and 2025, the Polito Regulation for inspection and defect classification, and technical datasheets for the installed materials. A unique panel code appears in both the BIM model and the inspection register. It links each panel's location to its condition history and serves as the shared asset identifier for controlled write-back.

The work was carried out in four stages. First, four locally executable LLMs were compared with expert-defined reference answers. Second, a retrieval-augmented agent was developed to select tools according to the task. Regulations and datasheets are handled through semantic retrieval, while the inspection register is queried computationally when an answer depends on exact records, filtering, calculation, or aggregation. Third, the selected prototype was tested with ten case-study questions, each repeated ten times, producing 100 answers. Fidelity to expert-defined answers and consistency across repeated runs were evaluated separately. An independent execution trace was also used to check which tools the agent actually called. Finally, a controlled write-back workflow was developed to transfer validated inspection findings to BIM. Because the corpus includes safety-related condition data, the entire pipeline runs locally and does not send project information to external inference services.

The prototype obtained a mean fidelity score of 0.708 and a mean self-consistency score of 0.879. Its answers were therefore more stable than they were accurate. Execution tracing also exposed limitations in the agent's ability to coordinate more than one tool within a single query. At its present level of performance, the system should be treated as a decision-support assistant rather than an autonomous authority. In the return workflow, validated findings are written through Dynamo to the instance parameters of the corresponding Revit elements, using the panel code as the key, and the revised data are exported back to the inspection register. No change is committed without human confirmation, while IFC property sets support interoperability. The thesis contributes a workflow for connecting heterogeneous maintenance information to BIM, an evaluation approach that separates accuracy from consistency and verifies actual tool execution, and a human-controlled method for keeping BIM condition information up to date under realistic institutional constraints.

2. Background and Literature Review

Over the past decade, Building Information Modelling (BIM) has developed from a digital modelling method used mainly for design coordination into an information environment intended to support the entire building lifecycle. This development is particularly important during operation and maintenance (O&M), where decisions depend on much more than geometry. Facility managers may also need asset identities, spatial locations, inspection histories, technical documents, system relationships, sensor readings, and information about current work orders. Large Language Models (LLMs) offer a new way to retrieve, interpret, and use this complex information. By combining LLMs with BIM, users could ask questions in ordinary language, access relevant model and document data, prepare maintenance actions, and support decisions without having to navigate specialised BIM software directly.

This chapter first considers BIM as a lifecycle information system and then reviews its use in O&M, maintenance planning, and interoperability. It subsequently introduces LLMs, agents, and Retrieval-Augmented Generation (RAG), before examining the emerging integration of LLMs and BIM for information retrieval, task planning, and maintenance support. The sources are not treated as equally central to the research. Studies that directly combine LLMs, BIM, and maintenance or O&M workflows form the core of the review, while research on RAG, natural language processing (NLP), embeddings, the Internet of Things (IoT), and broader AI-BIM applications provide methodological background. Duplicate uploads were excluded, and each source was considered only once.

2.1 BIM as a Lifecycle Information System

BIM is commonly defined as a digital representation of the physical and functional characteristics of a facility. In practice, however, it is more than a three-dimensional model. It provides a structured information environment that can support collaboration, design coordination, construction planning, operation, maintenance, and asset management throughout the building lifecycle. For maintenance purposes, its value lies in the information attached to building elements. In addition to geometry, these elements may contain parameters such as material, type, manufacturer, classification code, installation date, technical performance, inspection history, and maintenance requirements. This data-rich structure is what allows BIM to remain useful beyond the design and construction phases.

The O&M literature emphasises that the value of BIM during maintenance depends largely on whether the model was developed with asset management requirements in mind. Heaton, Parlikad, and Schooling

link the limited adoption of BIM in O&M to weak integration between BIM models and the asset information systems already used by organisations. Their method demonstrates how BIM data can be transferred to relational databases and used to support Asset Information Models (AIMs). However, this process is effective only when object classification and information requirements are defined from the beginning of the project [7]. This has a direct implication for an LLM-BIM maintenance workflow: a language model can retrieve and reason over maintenance information reliably only when the underlying BIM model contains consistent, structured, and meaningful asset data.

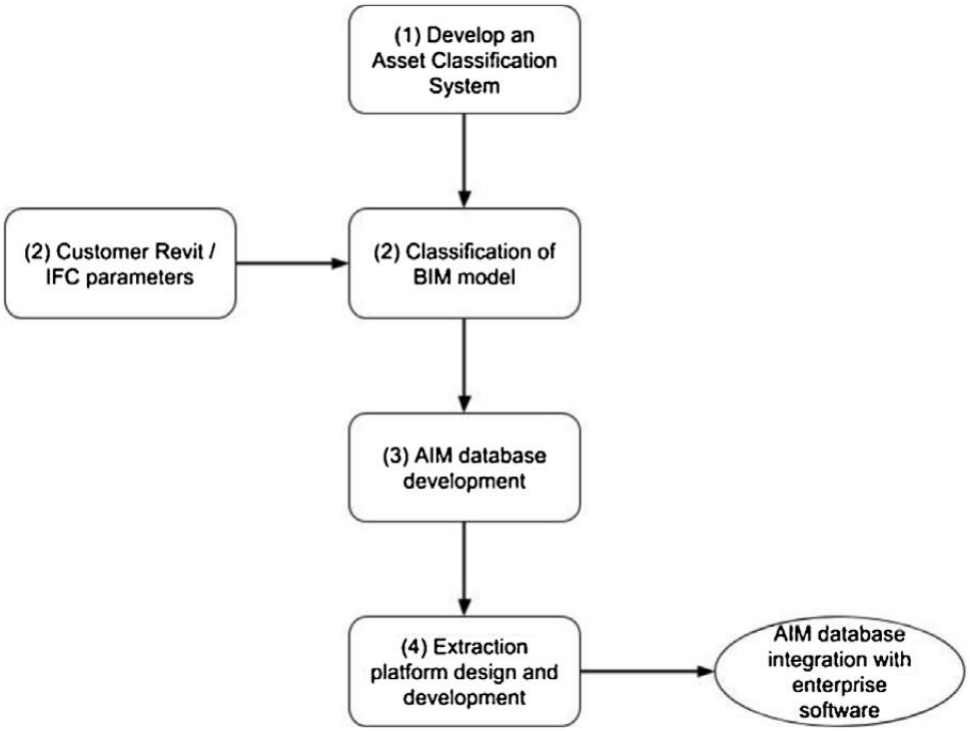


Figure 2.1. *BIM data extraction methodology for structuring asset information (Source: [7]).*

Interoperability is another recurring concern. A BIM-based maintenance workflow may involve several platforms, including Revit, Navisworks, facility management software, spreadsheets, databases, and a Building Management System (BMS). The Industry Foundation Classes (IFC) standard is important because it enables information to move between these systems without being restricted to a single software vendor. Inojosa and Vilanova demonstrate this approach for existing buildings by linking maintenance sheets and inspection records to structural elements in a 3D model and viewing the combined information in Navisworks Manage [8]. Their study presents a functional BIM maintenance workflow developed shortly before LLM-based interfaces became common. Elements, inspection data, maintenance sheets, and visual

information are connected, but users still need to navigate the software manually and know where the required information is stored.

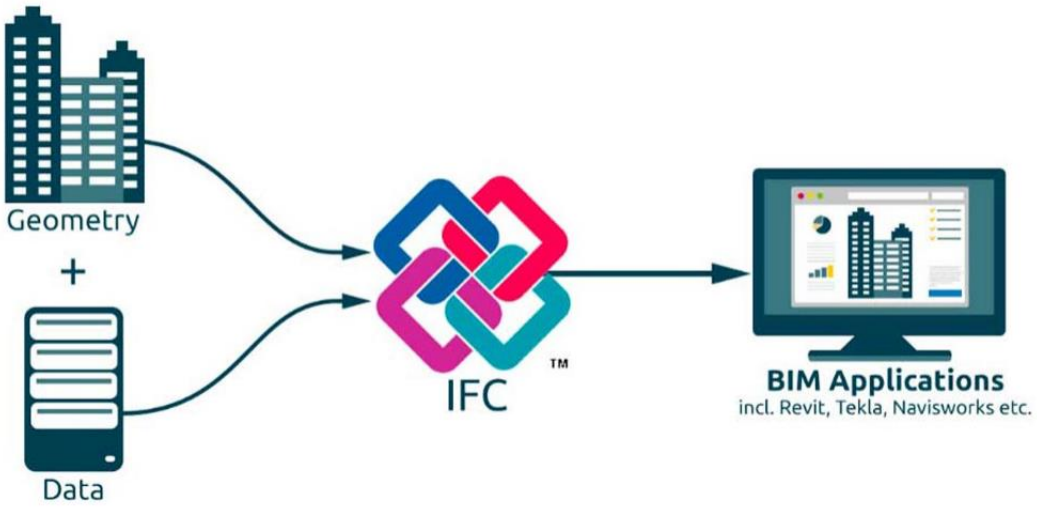


Figure 2.2. *Geometry and data combined through the IFC standard for exchange between BIM applications (Source: [8]).*

BIM-based O&M is also closely related to the concept of a digital twin. A digital twin extends BIM by connecting the virtual building to real-time or near-real-time operational data, usually through sensors, IoT devices, or a BMS. Ehab, Mahdi, and El-Helloty describe a BIM maintenance system integrated with IoT, in which model data and sensor data are combined to support decision-making, performance monitoring, and facility management [9]. This connection is important because maintenance involves more than retrieving static information. Buildings deteriorate, systems move away from their intended operating conditions, and faults may appear unexpectedly. An intelligent maintenance workflow must therefore connect stable BIM asset information with changing operational data.

2.2 BIM for Maintenance Planning, Facility Management, and Existing Buildings

The operation and maintenance phase is one of the longest and most costly stages of a building's lifecycle, yet BIM is less established in O&M than in design and construction. One reason is that O&M teams often inherit models created primarily for design coordination. The geometry may be accurate, while asset identifiers remain incomplete, system relationships are missing, property sets are inconsistent, and links to manuals or inspection records are weak or unavailable. As a result, facility managers may continue to rely on spreadsheets, paper records, manual inspections, or separate maintenance software even when a BIM model exists.

Heaton et al. also clarify what an O&M-oriented BIM model should contain. They argue that BIM must be organised around asset information requirements and object classification so that its data can be transferred to databases and existing asset management systems [7]. For the present thesis, this means that an LLM-BIM workflow should not be designed simply as a conversational interface added to an existing model. It requires a clear information structure that connects asset identifiers, spaces, systems, properties, documents, maintenance history, and work-order activities. Without this foundation, the natural-language interface may appear functional but will have little reliable information to retrieve.

These challenges are more pronounced in existing buildings, where documentation is often incomplete or outdated. Inojosa and Vilanova focus specifically on maintenance planning in this context and discuss the difficulties caused by missing historical drawings, undocumented interventions, and unstructured inspection data [8]. Their method links each structural element in the BIM model to its maintenance sheets and visually represents the identified problems. In this way, the model becomes a shared record of how the building's condition changes over time. With an LLM-based interface, users could query the same information directly, for example by asking which elements have recorded defects, which inspections are due, which components fail repeatedly, or what intervention is recommended in the maintenance record.

Xu's review reaches a similar conclusion, noting that the use of BIM in O&M is still developing, particularly in large commercial complexes where maintenance includes equipment management, space management, emergency support, and lifecycle data management [17]. Although the review is less detailed than more recent studies, it confirms the long-standing importance of integrated information management. The newer contribution of LLMs is their ability to reduce the interface barrier. Instead of requiring users to understand model navigation or database query languages, an LLM can interpret their intent and retrieve the information they need.

Another relevant direction is the integration of BIM with IoT and Building Management Systems. Ehab et al. show how BIM-IoT integration can improve maintenance and facility management by providing real-time information and decision support [9]. Zheng, Marie, and Kubler address a related problem: BMS metadata often varies between vendors and buildings, making the semantic tagging of data points a slow and largely manual task. Their BMS-RAG framework uses retrieval-augmented LLMs to classify point metadata from limited context and align it with ontology-based standards such as Brick [10]. Although the study is not centered on BIM, it is directly relevant to maintenance because many decisions depend on BMS points, equipment names, sensor tags, and operational readings. BIM identifies the physical assets

and spaces, while the BMS provides live signals. Connecting these sources through semantic tagging and retrieval is therefore an important step towards intelligent maintenance support.

2.3 From BIM Interoperability to Semantic Reasoning

A central challenge in BIM-based maintenance is the heterogeneous nature of the available information. The same asset may appear as a BIM element, a row in a spreadsheet, an entry in a facility management database, a point in a BMS, a section of a technical manual, and a subject in an inspection report. A maintenance workflow therefore requires semantic interoperability. The system must be able to recognise that these different representations refer to the same physical component, space, system, or maintenance action.

Lin and Ren's systematic review of BIM, the Semantic Web, and building knowledge reasoning frames interoperability as a reasoning problem rather than only a matter of file exchange [14]. Semantic Web technologies, ontologies, and linked data can make building information machine-interpretable, support regulatory and performance reasoning, and connect heterogeneous sources. The review also identifies several limitations, including fragmented semantics, misaligned ontologies, scalability constraints, and performance bottlenecks [14]. Each of these issues is relevant to an LLM-BIM maintenance workflow. An LLM may interpret natural language and generate queries, but its answers cannot be verified reliably unless they are based on structured and trustworthy data representations.

Recent LLM-BIM research increasingly uses IFC and BIM schemas as the structured foundation for natural-language interaction. Xue et al. propose a multi-agent framework for BIM information retrieval that maps natural-language questions to IFC constructs, including element classes, element-property entities, and relationships [4]. Liu and Chen's BIMCoder follows a similar approach by converting plain-language requests into structured BIM query strings that can be executed by a BIM platform [5]. Both studies show that the main challenge is not generating an answer in natural language but aligning informal user expressions with the formal structure of the model. For example, a user may ask for 'broken doors on the second floor', while the relevant information is distributed across IFC classes, property sets, spatial relationships, defect records, or a maintenance spreadsheet.

Semantic reasoning therefore provides the connection between BIM data and LLM-based interaction. Ontologies, IFC schemas, and knowledge graphs can guide the system towards the correct elements and relationships, while the LLM makes the information accessible by translating user questions into formal queries and presenting the results in plain language. A reliable maintenance workflow is therefore likely to be hybrid rather than based entirely on either an LLM or a BIM database. BIM provides structured asset

information, ontologies and schemas impose semantic constraints, RAG grounds the response in relevant sources, and the LLM manages language interpretation, reasoning, and interaction with the user.

2.4 LLMs, Agents, and RAG as Enabling Technologies

Current LLMs are based on more than a decade of research in neural language modelling, attention mechanisms, and transformer architecture. Vaswani et al. introduced the Transformer and established self-attention as a scalable approach to language processing [31]. BERT later demonstrated the value of deep bidirectional pre-training for language understanding [30], while Clark et al. analysed how individual attention heads capture different linguistic relationships [32]. These foundational NLP studies do not address BIM or maintenance directly, but they explain the technical basis of the LLM-BIM systems reviewed in this chapter.

A generic LLM, however, is not sufficient for maintenance applications. Maintenance questions often require current, project-specific, and source-grounded information that was not included in the model's training data. RAG addresses this limitation by combining language generation with external information retrieval. Klesel and Wittmann describe RAG as a means of connecting a conversational agent to an organisation's own data, thereby reducing hallucinations and improving access to information [25]. Khan et al. provide a practical account of developing RAG systems from PDF files, covering data collection, preprocessing, indexing, retrieval, and answer generation [20]. This is particularly relevant to maintenance, where information is frequently stored in manuals, regulations, inspection reports, specification sheets, and other PDF documents.

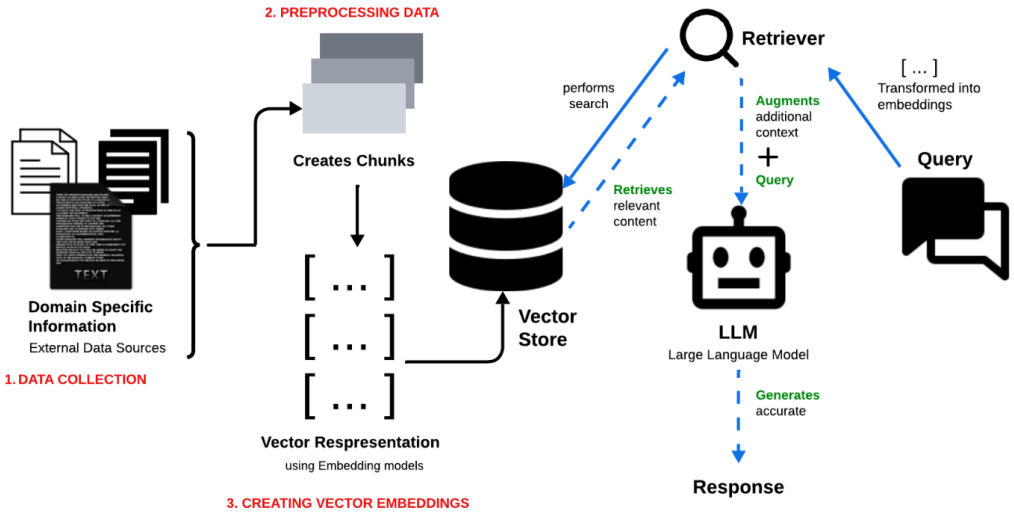


Figure 2.3. *Architecture of a Retrieval-Augmented Generation (RAG) system (Source: [20]).*

Several RAG studies are relevant to both the design and evaluation of a BIM maintenance assistant. Barnett et al. identify seven failure points and show how retrieval may fail when relevant content is missing, divided into unsuitable chunks, retrieved incorrectly, or not properly validated [22]. Gan et al. review RAG evaluation methods and argue that these systems should be assessed in terms of retrieval quality, answer faithfulness, factual accuracy, safety, and efficiency [21]. PaperQA presents a RAG agent that retrieves full-text scientific papers, evaluates their relevance, and produces answers grounded in citations [23]. HySemRAG extends this approach to automated literature synthesis and research-gap analysis through hybrid retrieval, knowledge graphs, and agent-based quality assurance [24]. These studies indicate that a thesis prototype should not be evaluated simply on whether its answers sound convincing. Evaluation must also determine whether the correct element, passage, or record was retrieved and whether the final response is supported by that evidence.

Embeddings and dense retrieval form the technical basis of many RAG systems. Dense Passage Retrieval demonstrates that vector representations can identify relevant passages even when there is little direct keyword overlap [26]. Sentence-BERT and the Universal Sentence Encoder provide practical methods for generating sentence embedding for semantic search and similarity comparison [27,29]. However, Steck et al. warn that cosine similarity in an embedding space does not always correspond to meaningful semantic similarity, particularly when embedding norms and regularisation effects influence the result [28]. This limitation is especially important in BIM maintenance, where terminology may be technical, abbreviated, and specific to one project. A purely vector-based search may confuse similar-looking terms or fail to

identify the correct component. Reliable retrieval therefore benefits from hybrid search, schema-based filters, and domain-specific metadata.

Agent-based interfaces provide another enabling technology. Chin, Wang, and Xia describe an LLM agent as an interface between a user and a complex system. In this role, the agent moves beyond passive command following by helping users clarify their needs and then activating the appropriate tools [19]. This is well suited to maintenance, because a facility manager or technician may not know which BIM query or software command is required to obtain an answer. An agent could ask for clarification, determine whether the request concerns asset data, a manual, an inspection record, or a task plan, and then call the appropriate BIM, database, or RAG tool on the user's behalf.

2.5 Existing BIM-LLM Integration in AECO Workflows

Park et al. provide the broadest overview of BIM-LLM integration through a review of 61 studies across Architecture, Engineering, Construction, and Operations (AECO) [3]. They classify the field into information retrieval, data modelling, compliance checking, performance optimisation, construction information interaction, and knowledge modelling. This classification is useful for positioning the present thesis because maintenance lies at the intersection of information retrieval, knowledge modelling, and lifecycle operation. Park et al. also note that many BIM-LLM studies include moderate technical validation but limited industrial validation [3]. In other words, a system may perform well in a controlled experiment without having been tested in a real maintenance environment.

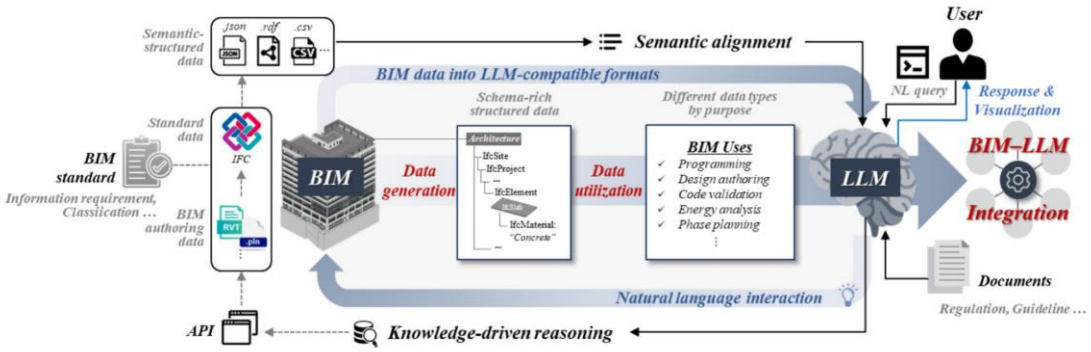


Figure 2.4. *Semantic interoperability framework for BIM-LLM integration (Source: [3]).*

Within this literature, natural-language information retrieval is particularly relevant. Xue et al. propose a multi-agent collaboration framework that adapts its agent workflow to different BIM queries and aligns questions with IFC classes, properties, and relationships [4]. Their results are strong for query classification and relationship alignment, while property alignment remains more difficult. This limitation is important in O&M because many maintenance questions concern properties such as condition status, installation

date, equipment capacity, warranty period, failure history, and the date of the last inspection. Weak property alignment may therefore allow the system to identify the correct element while returning the wrong maintenance attribute.

Liu and Chen's BIMCoder also focuses on natural-language BIM information retrieval. It translates a user's request into a structured BIM query statement and acts as an intermediate layer between non-expert users and professional BIM platforms [5]. This directly addresses a key requirement of a maintenance assistant: facility managers should be able to obtain information from the model without understanding its internal schema or query language. BIMCoder suggests that fine-tuned or combined LLM approaches can perform this translation, although the framework is designed for general BIM retrieval rather than maintenance-specific tasks.

Nithyanantham et al.'s MCP4IFC extends LLM-BIM interaction from querying to editing IFC data through tool calls [6]. Using the Model Context Protocol, the framework allows the model to query, create, and modify IFC content through BIM tools, predefined functions, RAG, and code generation. This is relevant to maintenance because a future workflow may need to do more than return an answer. It may also annotate an element, update an asset's condition, create a work order, or change an inspection status. At the same time, allowing an LLM to modify building data introduces risks related to control, validation, and safety. A maintenance workflow should therefore clearly separate read-only retrieval, recommendations, and authorised updates to the model.

Several other LLM-BIM studies provide useful comparisons, although they are not focused on maintenance. Madireddy et al. combine LLMs with Revit to interpret building codes, generate Python scripts, and perform semi-automated compliance checks [11]. Conceptually, this is similar to converting maintenance procedures into checks on specific assets. Lee, Jang, and Hyun propose a generalised LLM-augmented BIM framework for speech-to-BIM interaction, organised around interpretation, matching, execution, and checking [12]. Sketch2BIM demonstrates multi-agent human-AI collaboration for converting hand-drawn floor plans into BIM models [15], while Bagasi, Nawari, and Alsaffar examine the use of BIM and AI in early-stage design communication [16]. Although these studies address different tasks, they reflect a common direction: LLMs are increasingly used as natural-language interfaces between users and BIM processes, including querying, model generation, design communication, and rule checking.

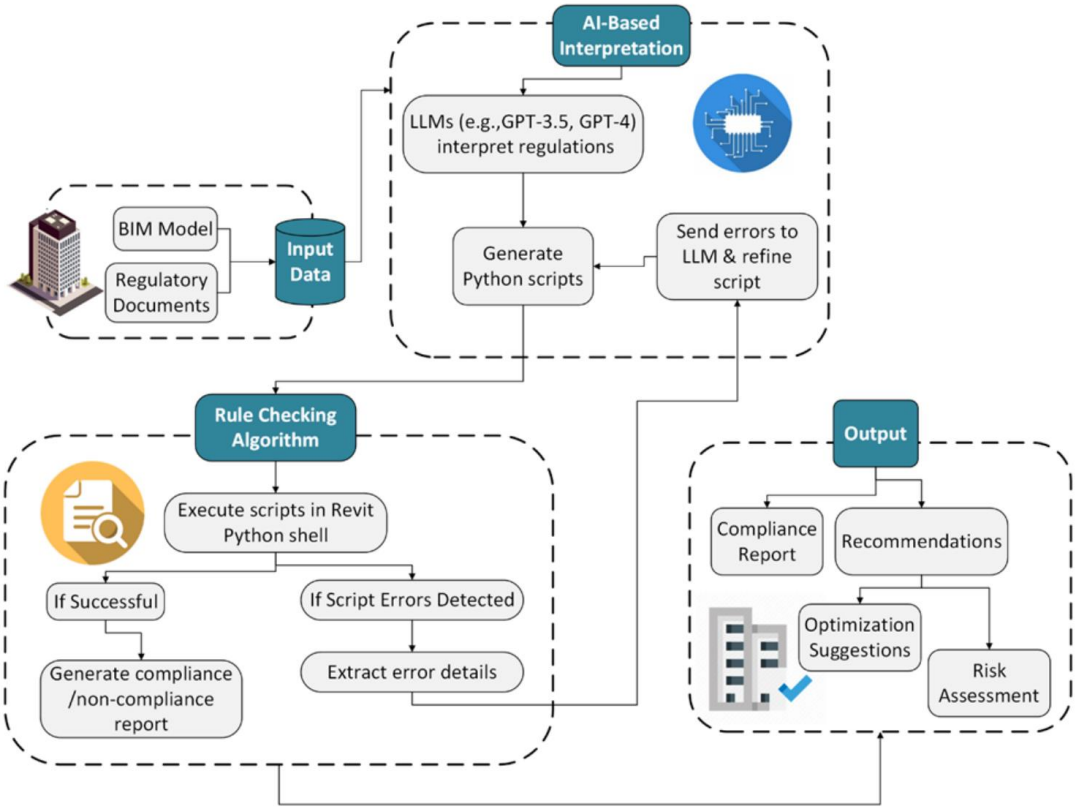


Figure 2.5. *LLM-based framework for automated code-compliance checking in BIM (Source: [11]).*

2.6 LLMs and BIM for Maintenance Workflow

The study most directly related to this thesis is Kim's work on automated robotic planning for corrective building maintenance using LLMs and BIM-driven 3D scene graphs [1]. The research addresses corrective maintenance, in which an unexpected defect requires a rapid response, and proposes LLM-based modules for semantic search, robot selection, navigation planning, scanning planning, and task planning. Its central component is the BIM-driven 3D Scene Graph (BIM3DSG), which transforms BIM information into a more compact representation suitable for robotic reasoning and reduces the amount of information that must be processed within the LLM's token limits [1]. The study is particularly relevant because it combines BIM data, LLM reasoning, and a maintenance workflow. It also extends the problem beyond information retrieval by considering how an action should be planned once the necessary information has been identified.

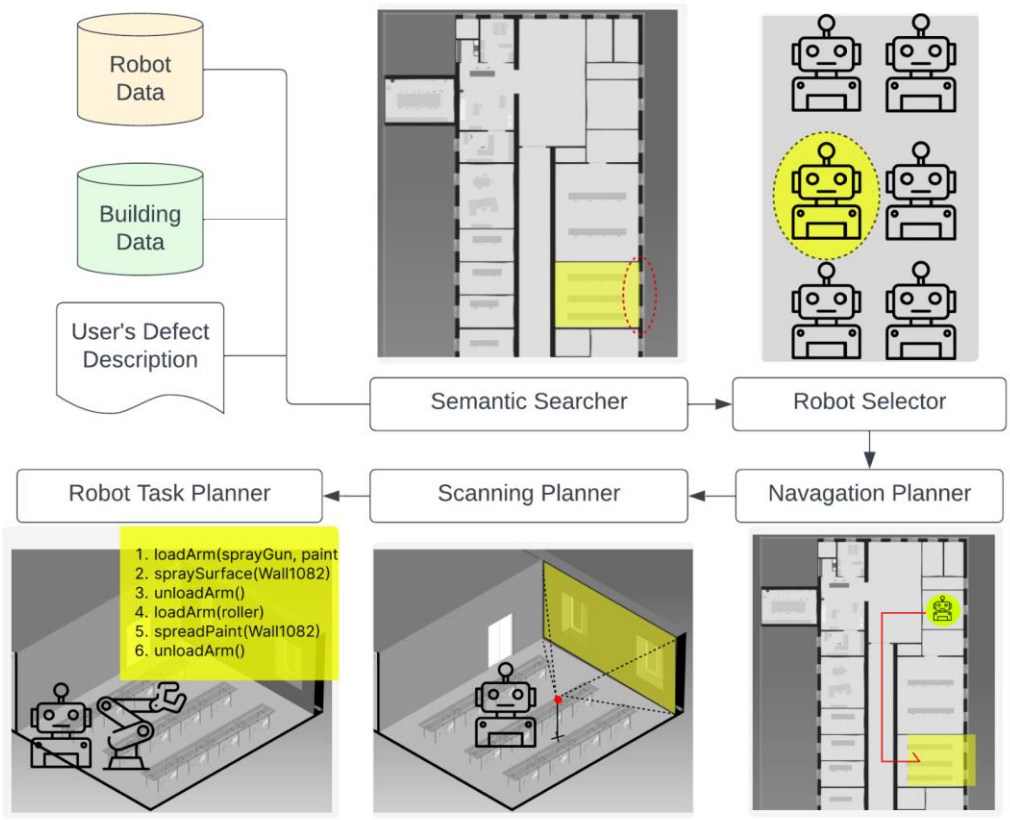


Figure 2.6. *LLM- and BIM-driven framework for corrective maintenance task planning (Source: [1]).*

Kim's framework also highlights the importance of intermediate representations. A BIM model contains extensive detail, but its raw structure may not be suitable for direct reasoning by an LLM or a robot. A scene graph can represent rooms, objects, relationships, and navigable areas in a more compact form. The same principle applies beyond robotics. An LLM-BIM maintenance workflow may require its own intermediate representation to connect elements, spaces, maintenance records, manuals, sensors, and user tasks. Such a layer can reduce the amount of raw BIM data passed to the LLM and provide tighter control over the reasoning process.

Li et al. provide the second core study through an LLM-based information-query paradigm for building O&M [2]. Their pipeline includes data processing, fine-tuning of an O&M-specific LLM, knowledge enhancement through an external O&M knowledge base, RAG, a knowledge-graph search algorithm, and alignment fine-tuning based on human feedback. The study is directly relevant because information querying is an essential part of any maintenance workflow. Although it is not strictly a BIM-focused paper,

it addresses the same practical challenge: O&M data are complex, heterogeneous, and difficult to query through conventional database methods. Its use of RAG and knowledge graphs therefore provides a useful methodological reference for this thesis.

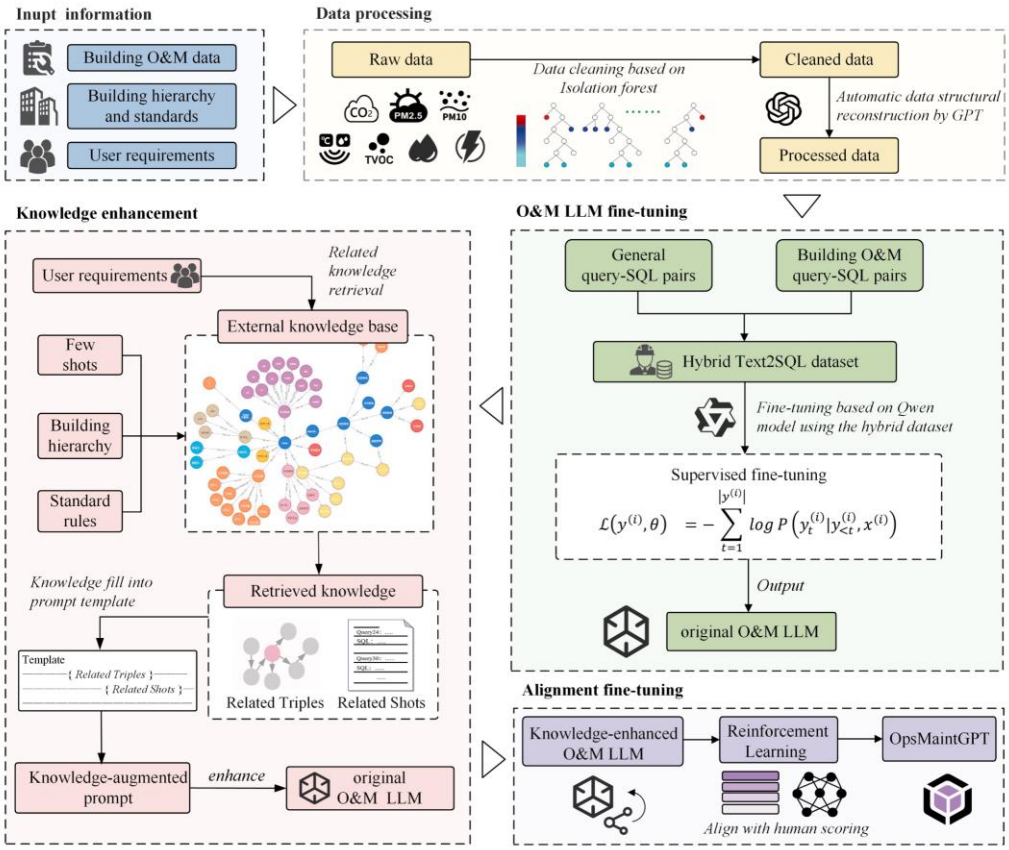


Figure 2.7. *LLM-based building O&M information-query workflow (Source: [2]).*

Taken together, Kim and Li et al. describe two complementary levels of maintenance intelligence. The first is information querying: locating and explaining relevant O&M data, manuals, sensor readings, or asset records [2]. The second is task planning: deciding what action is required, which tools or robots are needed, and how the physical work should be carried out [1]. A complete LLM-BIM maintenance workflow is likely to require both levels. For example, when a user reports a defective component, the system could first locate the asset in BIM and retrieve its position, properties, maintenance history, and manual instructions. It could then propose a task plan, inspection route, or work order.

Akhavian et al.'s BIM2RDT is an ambitious related framework that combines agentic AI, BIM, digital twins, robotics, and safety [13]. Its main focus is construction-site safety and robot-ready digital twins rather than building maintenance. Nevertheless, it demonstrates how BIM information, live sensing, robot-collected

data, and LLM reasoning can be integrated. The framework uses a feedback loop in which observations collected by robots update the digital twin, and the updated twin then informs the next mission [13]. This principle is directly relevant to maintenance: inspection findings should update asset-condition records, and the updated records should influence subsequent maintenance decisions.

IoT and BMS integration completes this broader picture. Yang and Su discuss how LLMs may support Social IoT applications such as recommendation, search, and data management [18]. Although their work is not focused on BIM, it reinforces the role of LLMs as an interface between people and connected devices. Zheng et al. provide a building-specific example by using RAG for BMS point tagging and ontology compliance [10]. When considered alongside BIM-IoT maintenance research [9], these studies suggest that a future LLM-BIM maintenance assistant should not rely only on static BIM geometry. It should also be able to use O&M documents, BMS metadata, sensor data, inspection records, and potentially live digital-twin updates.

2.7 Human Interaction and Maintenance Workflow Design

Maintenance is inherently a socio-technical process. It involves facility managers, technicians, owners, inspectors, software systems, asset databases, and, in some cases, robots or other automated devices. The success of LLM-BIM integration therefore depends not only on retrieval accuracy but also on the quality of the interaction between users and the system. Users often describe problems in non-technical language, omit relevant details, or identify assets by location rather than by a formal code. A technician, for instance, is more likely to report that 'the air unit near the east corridor is noisy' than to provide its asset ID. The system must interpret this description, request clarification when necessary, and still retrieve the correct BIM or O&M information.

Chin et al.'s human-centered LLM-agent interface supports the idea of using an LLM as the link between the user and the underlying system [19]. In a maintenance setting, this interface should do more than answer isolated questions. It should help guide the workflow by interpreting the issue, locating the relevant asset, retrieving documents, reviewing its history, recommending the next step, preparing a report, and updating records after human approval. The LLM should therefore act as an orchestrator and decision-support assistant rather than as an independent authority.

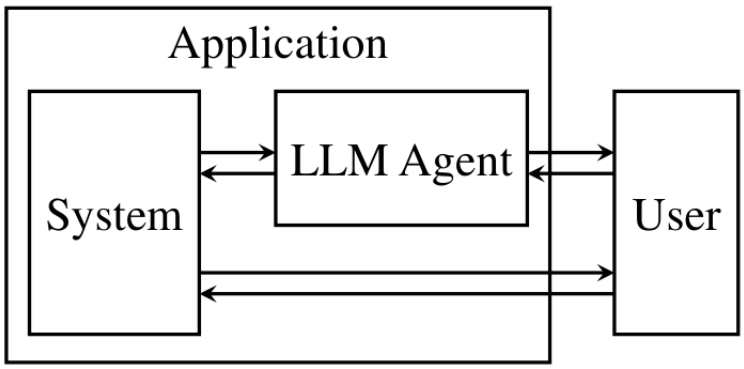


Figure 2.8. *The LLM agent as an interface between the user and the underlying system (Source: [19]).*

The literature also shows that automation can be introduced at different levels. Some low-risk tasks, such as retrieving asset properties or summarising a manual, may be automated safely. Higher-risk actions, including confirming a diagnosis, modifying model data, scheduling work, or dispatching a robot, require human approval. Madireddy et al.'s code-compliance framework provides a useful example of semi-automation: the LLM interprets regulations and generates actions, while execution in the BIM environment and the validation stage remain controlled [11]. Maintenance requires a similar approach, in which the LLM supports technicians and facility managers without replacing their professional judgement.

Speech-to-BIM and natural-language BIM frameworks make a related point about checking and validation [12]. In a maintenance context, the system should maintain a traceable record of the evidence used, communicate uncertainty, and allow users to correct incorrect assumptions. This reflects an important lesson from RAG research: these systems are not automatically robust, and their reliability must be developed through validation, feedback, and continued operational monitoring [22].

2.8 Evaluation, Validation, and Limitations

Validation remains a major weakness in BIM-LLM research. Park et al. report that many studies do not progress to real-world deployment [3]. This limitation is particularly serious in maintenance, where an incorrect answer may lead to wasted labour, an undetected defect, a safety risk, or an inappropriate intervention. An LLM-BIM system should therefore be evaluated on two related levels: the quality of information retrieval and the performance of the overall maintenance workflow.

The RAG evaluation literature provides several metrics for the first level, including retrieval precision and recall, answer faithfulness, citation accuracy, robustness, latency, and user satisfaction [21]. In BIM maintenance, these metrics can be translated into practical questions. Did the system identify the correct building element? Did it retrieve the appropriate property, document, or inspection record? Did it

understand the user's maintenance request? Was the response safe and actionable? Did it provide enough evidence for a facility manager to verify the answer? Finally, did the workflow reduce the time required compared with a manual search through BIM data and documents?

Barnett et al.'s failure points apply directly to this context [22]. A maintenance RAG system may fail because a relevant document was not ingested, because the chunking process separated a procedure from its warning, because vector search returned a similar but incorrect item of equipment, or because the model generated a plausible answer without adequate evidence. These risks are amplified in BIM environments, where asset names, room numbers, and equipment tags may be ambiguous. Steck et al.'s warning about cosine similarity is also relevant: proximity in an embedding space does not necessarily indicate correct technical similarity [28]. Evaluation should therefore include schema-based checks, element-ID matching, and review by human experts rather than relying on answer fluency alone.

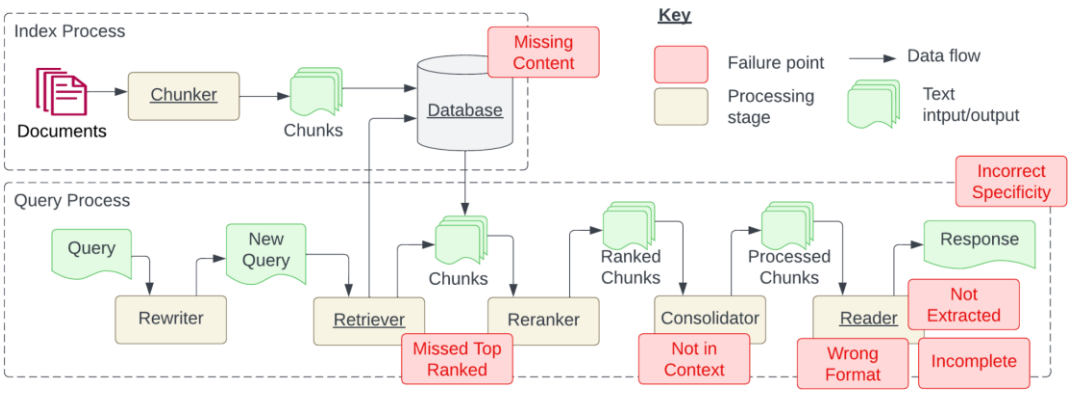


Figure 2.9. *RAG indexing and query workflow with seven potential failure points (Source: [22]).*

Foundational research on retrieval and embeddings supports the technical design while also showing why a system must be adapted to its domain. Dense Passage Retrieval and sentence embeddings provide effective methods for semantic search [26,27]. The t-SNE method may also be used, within a limited scope, to visualise document or embedding clusters [33]. Bayesian optimisation and more advanced LLM search strategies are less central to the present study, although they may become relevant in future work for tuning retrieval parameters or applying planning-based search [34,35]. They are therefore considered supporting rather than core methods in this thesis.

2.9 Research Gap and Positioning of the Thesis

The existing literature has developed along three main lines. BIM-O&M studies show how BIM data can support asset management, inspection records, maintenance planning, and facility management [7–9,17]. Research on LLMs and RAG explains how language models can retrieve, synthesise, and present domain-

specific information from external sources [20–25]. BIM-LLM studies demonstrate that language models can support BIM information retrieval, IFC querying, model editing, code checking, and natural-language interaction [3–6,11–16]. However, these three lines of research have not yet been fully combined into a practical maintenance workflow.

The closest existing contributions are Kim's LLM-BIM framework for corrective-maintenance task planning [1] and Li et al.'s LLM-based O&M information-query paradigm [2]. Even with these studies, there remains a clear opportunity to examine the integration workflow itself: how BIM data, O&M documents, maintenance records, and natural-language interaction can be connected within one coherent maintenance-support system. Current BIM-LLM retrieval studies are often not maintenance-specific [4,5]. BIM-O&M research generally does not include an LLM-based interface [7–9], while RAG research is usually domain-general and does not address BIM element structures or maintenance workflows [20–25].

The central research gap can therefore be stated as follows: although previous studies have demonstrated the potential of BIM for O&M and the value of LLMs for natural-language information retrieval, there is still limited practical guidance on integrating the two around BIM-based asset information in a structured, traceable, and user-centered workflow. This broad gap includes several more specific issues: aligning everyday language with BIM and IFC data; combining static BIM asset information with dynamic O&M or BMS data; retrieving information reliably from technical documents and maintenance records; validating the complete workflow rather than only individual question-answer pairs; and maintaining human control over maintenance decisions.

Against this background, the thesis can contribute by proposing and evaluating a workflow architecture for LLM-BIM maintenance support. The architecture may combine: (1) BIM/IFC asset extraction and classification; (2) an O&M knowledge base containing manuals, inspection records, and maintenance documents; (3) RAG-based retrieval supported by semantic search and metadata filters; (4) an LLM-agent layer that interprets queries and coordinates tools; (5) a response layer that provides source-grounded answers, asset references, and suggested actions; and (6) a validation strategy based on expert review, retrieval accuracy, and workflow usability. This direction is strongly supported by the most relevant literature while maintaining a clear and distinct scope for the thesis.

3. Case Study: The Dumarey Building Facade

3.1 Why this building was chosen

Demonstrating a maintenance-support framework requires more than a hypothetical example. It needs a real asset, a reliable record of its condition, and a maintenance issue that the system can address in practice. The Dumarey building at the Politecnico di Torino meets these requirements. A federated BIM model of the envelope is available, each facade panel has a unique identifier shared by the model and supporting records, and the panels have been monitored through a nine-year inspection programme. For this reason, the building is not used simply as an illustration; it provides the data on which the evaluations in Chapters 4 and 5 are based.

Although the documentation is unusually complete, the maintenance problem itself is not unusual. Many institutional and commercial buildings have large, repetitive cladding systems that age over time and must be inspected element by element under a formal maintenance regime. The value of the Dumarey case lies in the combination of a representative problem and a well-structured evidence base. Its mix of BIM data, inspection records, and technical documents also reflects the heterogeneous information environment for which the architecture in Chapter 4 was developed.

3.2 Location and urban context

The building is located within the Cittadella Politecnica, the main campus of the Politecnico di Torino, in the Cenisia district west of central Turin. It faces Corso Castelfidardo, where several of the university's expansion buildings and overpass structures are situated. The premises at Corso Castelfidardo 36 are occupied by Dumarey Automotive S.p.A.; this association is the origin of the working name used throughout the thesis.

The setting is relevant to the maintenance problem for two practical reasons. The facade overlooks a busy public street, with pedestrian areas directly below the stone cladding. A loosened or detached panel therefore creates a public-safety risk as well as a maintenance issue, which explains why the governing regulation discussed in Section 3.7 requires immediate isolation of the area when a severe defect is identified. In addition, the building remains active institutional and commercial use. Inspection and replacement work must therefore be planned while the building continues to operate, which is the type of real operational constraint the proposed framework is intended to support.



Figure 3.1. *The Dumarey building on Corso Castelfidardo, Turin, showing the Balmoral granite ventilated facade.*

3.3 The building and its ventilated Balmoral facade

The building envelope is formed by a ventilated facade clad with red Balmoral granite, a material that gives several of the Politecnico di Torino buildings along Corso Castelfidardo their recognisable appearance. In this construction, the stone slabs form an external skin supported by a metal substructure and separated from the structural wall by a continuous air cavity. The cavity helps manage heat and moisture, but it also means that the panels depend entirely on concealed fixings and framing for support. That dependency is central to the maintenance problem described in Section 3.4.

The cladding is divided into individual panels, and inspection, grading, and maintenance are all carried out at panel level. The BIM model follows the same logic by representing every panel as a separate element. This makes it possible to associate each part of the facade with its own condition record and to query the envelope panel by panel rather than treating it as a single surface.

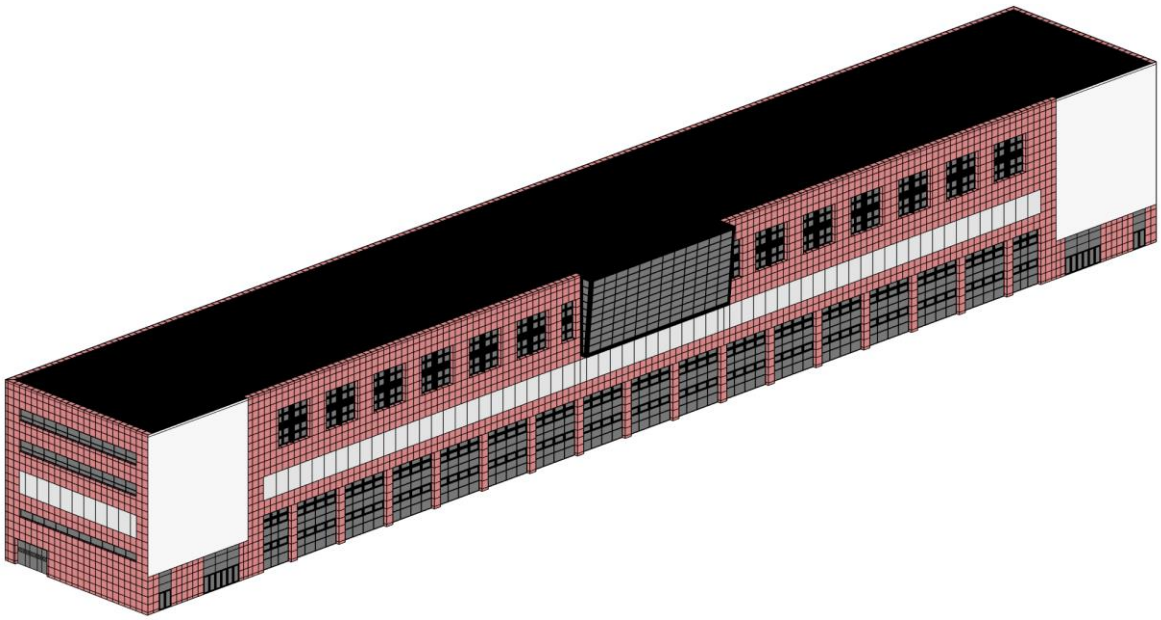


Figure 3.2. *Isometric view of the Dumarey BIM envelope model, with individual curtain-panel elements and facades A–D.*

Figure 3.2 shows the building as a long, low volume with two principal elevations along its length and two shorter end elevations. The stone panels form a regular grid, interrupted by horizontal window bands and by doors and vehicle-access openings at ground level. Colour coding in the model distinguishes the cladding panels from the glazed and framed elements. The same geometric organisation is reflected in the inspection register described in Section 3.6.

3.4 The maintenance problem

Over time, the Balmoral facades of the Politecnico buildings on Corso Castelfidardo have experienced deterioration of the metal substructure supporting the stone cladding. As concealed fixings and framing components degraded, some granite slabs became loose and, in the most serious cases, failed. This is particularly concerning because many panels are positioned several metres above a public pavement. The condition of the facades has therefore led to a wider refurbishment programme involving the repair and replacement of natural-stone cladding across the affected buildings.

The inspection data used in this thesis records the development of this real maintenance problem. For each panel and inspection period, the register shows how the assigned defect grade changes as the condition evolves and how the required action progresses from observation to replacement once the severe threshold is reached. The 150 panels that received the highest grade, discussed in Section 3.6, correspond to the failures that prompted physical intervention. The case study is therefore based on the

documented history of an actual, safety-relevant maintenance programme rather than on a constructed scenario.

3.5 The BIM model as the geometric backbone

The federated BIM model provides the geometric and semantic reference for the case study. It serves three main purposes in this research. First, each facade panel is represented as a separate element in the Curtain Panels category, with a defined location and set of attributes. Second, each element contains the alphanumeric identifier used to connect the different data sources, as explained in Section 3.8. Third, the model is the destination of the write-back process developed in Chapter 4, where validated inspection information is added to the corresponding panel elements as parameters.

At the start of the study, the model contained the as-built geometry and type information for the envelope, while the inspection history remained in a separate register. The methodology addresses this separation by allowing the current condition of each panel to be written back to the BIM model. In this way, the model can represent not only the designed asset but also its condition during operation.

3.6 The Panel Inspection Register

The Panel Inspection Register is a structured spreadsheet containing 5,496 panel records and 21 substantive columns. It is the most detailed source in the case-study corpus and provides the basis for most of the project-specific questions used in the system evaluation.

3.6.1 Panel taxonomy and facades

All panels belong to the Pannello sistema family within the Curtain Panels category. They are divided into two types according to thickness: Lastra Balmoral sp. 3 cm and Lastra Balmoral sp. 4 cm. The envelope is organised into four facades, labelled A to D, and the first character of each panel code indicates the facade on which the panel is located.

Table 3.1. *Distribution of the 5,496 panels across the four facades*

Facade	Panels	Share
A	2,398	43.6%
B	1,815	33.0%
C	629	11.4%
D	654	11.9%

Facades A and B form the two long elevations and together contain more than three-quarters of all panels. Facades C and D correspond to the shorter ends of the building. This distribution is consistent with the geometry visible in Figure 3.2.

3.6.2 Inspection history

For every panel, the register contains defect grades for twelve inspection periods on a scale from 0 to 4. Inspections were carried out twice a year in 2017 and 2018, annually from 2019 to 2024, and twice again in 2025. This pattern reflects periods of more intensive monitoring at the beginning and end of the available record. Across 5,496 panels and twelve periods, the dataset contains 65,952 individual grade observations. Two additional fields summarise each panel's record: Max Grade Recorded, which stores the highest grade observed at any time, and Required Action (current), which describes the action associated with the most recent condition.

These two summary fields describe different aspects of the same asset and should not be read as duplicates. A panel that received Grade 4 in an earlier inspection may have been replaced and may now require no further action. In that case, Max Grade Recorded correctly remains 4, while Required Action (current) may state No action. One field preserves the historical maximum; the other describes the present maintenance status. This distinction was explicitly incorporated into the framework because overlooking it can lead to incorrect answers.

3.6.3 Condition of the envelope

Most panels have only minor recorded defects. This distribution is consistent with an envelope that was still relatively early in its service life when the inspection record began.

Table 3.2. *Distribution of worst-observed defect grade (Max Grade Recorded)*

Grade	Severity (Regulation §9)	Panels
0	No defect	11
1	Minor cosmetic defect	5,242
2	Minor functional defect	23
3	Significant defect	70
4	Severe / safety-critical (replacement)	150

Two values in Table 3.2 are used repeatedly in the evaluation and must be distinguished carefully. A total of 150 panels reached Grade 4 at least once during the recorded period; these are the panels that required replacement at some point. By contrast, 50 panels have replacement listed as the current required action because they were graded 4 in the final inspection period. The first figure answers the historical question,

'How many panels have ever failed?', whereas the second answers the current operational question, 'How many panels require replacement now?' Both values are correct, but they refer to different time frames. This distinction is examined again in Chapter 5.

3.7 Supporting documentation

Two additional groups of documents complete the case-study corpus. Unlike the inspection register, both are primarily textual and are therefore handled as unstructured sources.

The Polito Regulation defines the procedures for facade-panel inspection. Its sixteen numbered sections cover inspection policy and frequency, inspection methods, technical checklists, defect classification, maintenance and replacement requirements, responsibilities, and safety. Section 9, Defect Classification and Risk Levels, defines the 0–4 grading scale used in the register. It therefore provides the link between the recorded values and their operational meaning: a grade assigned to a panel can only be interpreted correctly in relation to the requirements set out in that section.

The technical datasheets describe the products installed in the facade system, including the HILTI HST anchors used for the substructure, ISOVER rock-wool insulation, and Neopor graphite-enhanced polystyrene. These documents provide manufacturer-based information for questions concerning the materials and components of the envelope.

Table 3.3. *The case-study corpus and how each part is handled*

Source	Nature	Role in the framework
BIM model	Geometry and attributes	Panel identity and write-back target (Chapter 4)
Panel Inspection Register	Structured table (5,496 panels)	Condition data; queried computationally
Polito Regulation	Prose (16 sections)	Rules and grade definitions; retrieved semantically
Material datasheets	Prose	Material properties; retrieved semantically

3.8 The panel code: the key that unifies the case study

A single identifier connects the main sources in the case study. Each panel has a unique alphanumeric value stored in the C_Codice Lastra field, such as AL3G16F15. The same value appears in the BIM model and the inspection register and is also used in the retrieval and write-back workflow developed in Chapter 4. As a result, a natural-language question about one panel can be linked to its geometric location, inspection history, and the regulation used to interpret its condition.

Direct checks of the register confirmed three properties that make the code suitable as a join key. First, all 5,496 codes are unique, so each lookup identifies one panel only. Second, the codes do not all have the same length: most contain nine characters, while 176 contain ten or eleven. Matching must therefore use the complete string rather than fixed character positions. Third, the first character consistently identifies the facade, using A, B, C, or D. Chapter 4 uses this redundant information as a validation check during write-back. These properties are established here because the reliability of the later integration depends on them.

3.9 Suitability of the case study

The Dumarey facade is particularly suitable for this study for four related reasons.

First, it is based on a real and safety-relevant maintenance problem. The dataset records deterioration in a stone facade located above a public pavement, so the framework is evaluated in a context where incorrect information could have practical consequences.

Second, the available information is heterogeneous in the same way as the problem addressed by the methodology. A large, structured register must be used alongside regulatory and technical documents, and the sources are connected through a common panel identifier. This combination directly motivates the two-tool architecture introduced in Chapter 4.

Third, the case-study material is sufficiently complete to support the full workflow. The BIM model, condition records, governing rules, and material specifications are all available, allowing both retrieval and write-back to be considered within the same study.

Finally, the case is representative of a wider class of maintenance problems. Large, repetitive, panelised cladding systems are common in institutional and commercial buildings, and their condition must often be tracked over time under a formal inspection regime. The lessons drawn from the Dumarey facade may therefore be relevant to comparable assets.

4. Methodology

4.1 Research design

This chapter explains how the proposed framework was designed, implemented, and evaluated. The work is based on the case study presented in Chapter 3: the Dumarey building, its BIM model, the Panel Inspection Register, and the applicable regulation. The data are therefore referenced here without being described again.

The study develops and evaluates an interactive framework for two-way information exchange between a facility manager and a BIM-based asset database, with a Large Language Model (LLM) acting as the interface. From the outset, the full pipeline was required to run locally. The case-study dataset records the actual condition of a university building, including safety-critical defects, so sending it to a third-party inference API was considered unacceptable. This decision also restricted the available models to the 4B-7B range and influenced many of the design choices discussed in this chapter.

The research was organised into four phases. The output of each phase informed the design of the next.

Table 4.1. *Structure of the research*

Phase	Objective	Principal output
1. Model selection	Identify which locally-runnable LLM best represents BIM/FM domain knowledge	Ranked benchmark of four candidate models
2. System construction	Build a retrieval-augmented, tool-using agent over the case-study corpus	bim_fm_rag software prototype
3. Evaluation	Quantify answer fidelity and answer stability of the prototype	Fidelity and self-consistency heat maps
4. BIM integration	Return inspection outcomes to the BIM model, the documentation and the retrieval corpus	Inspection parameter schema and Dynamo exchange graphs

Phases 1 to 3 have been completed and are reported together with their results. Phase 4, which covers BIM integration, is presented in Section 4.5 through the implementation and demonstration of the Dynamo exchange.

The framework supports two directions of information flow. In the read direction, the facility manager queries the building documentation and inspection register in natural language. In the write direction, findings from new inspections are converted into structured records and returned to the BIM model, after

which they become available to the retrieval layer. Phases 1 to 3 address the read path, while Phase 4 addresses the write path.

4.2 Comparative selection of the local LLM

4.2.1 Objective and candidate models

A base model had to be selected before the retrieval system could be developed. The initial comparison focused on each model's existing knowledge of BIM and facility management, including its ability to understand the vocabulary and concepts of the field before any project document was retrieved. This was important because successful retrieval alone is not sufficient if the model misinterprets the retrieved material.

Four openly available models, all runnable locally through Ollama, were compared.

Table 4.2. *Candidate models*

Model	Parameters	Notable characteristic
deepseek-r1:7b	7B	Reasoning-distilled
mistral:7b	7B	General-purpose, tool-calling capable
gemma3:4b	4B	Compact, Google
qwen3:4b	4B	Compact, hybrid reasoning, native tool calling

4.2.2 Question set and reference answers

A set of twenty BIM and facility-management questions were prepared. It included definitions, such as "What is Building Information Modelling?" and "What is COBie and how is it used in FM?"; applied questions, such as "How does BIM improve maintenance planning?"; and integration questions concerning digital twins and IoT. An expert reference answer was written for each question and used as the scoring baseline. The complete list of the twenty questions is given in Appendix A.

Each question was submitted to all four models under identical conditions, using a single fixed decoding configuration and the same local runtime. This produced 80 responses, which were stored in 20 Questions - all answers.xlsx.

4.2.3 Scoring by embedding similarity

Exact-match scoring is unsuitable for free-text answers because equivalent ideas can be expressed in different words. Semantic similarity was therefore used. Each model answer and its expert reference

answer were converted into dense vectors using embeddinggemma:300m, running locally through Ollama, and compared by cosine similarity:

$$sim(a, r) = (v_a \cdot v_r) / (\|v_a\| \|v_r\|)$$

where v_a and v_r represent the embeddings of the model answer and the reference answer. For the vectors used in this study, the score lies between 0 and 1, with higher values indicating greater semantic similarity.

The prompt format used for the embedding model is important to the validity of these scores. EmbeddingGemma is trained with task-specific prefixes, and omitting the prefix can produce well-formed vectors that are not meaningful for the intended comparison. Because this is a symmetric sentence-similarity task, both texts were prefixed with task: sentence similarity | query: before embedding. The same convention was used for all scores reported in Sections 4.2 and 4.4.

The 20 × 4 matrix of scores was then visualised as a heat map, allowing both overall performance and question-level differences to be examined.

4.2.4 Results and model selection

Table 4.3. *Model selection benchmark results (cosine similarity to expert reference answers, 20 questions)*

Model	Mean similarity	Std. dev.	Min	Max	Questions won (of 20)
qwen3:4b	0.8284	0.0805	0.5769	0.9192	8
mistral:7b	0.8203	0.0708	0.6424	0.9059	5
gemma3:4b	0.8189	0.0712	0.6293	0.9027	6
deepseek-r1:7b	0.7973	0.0739	0.5858	0.8764	1

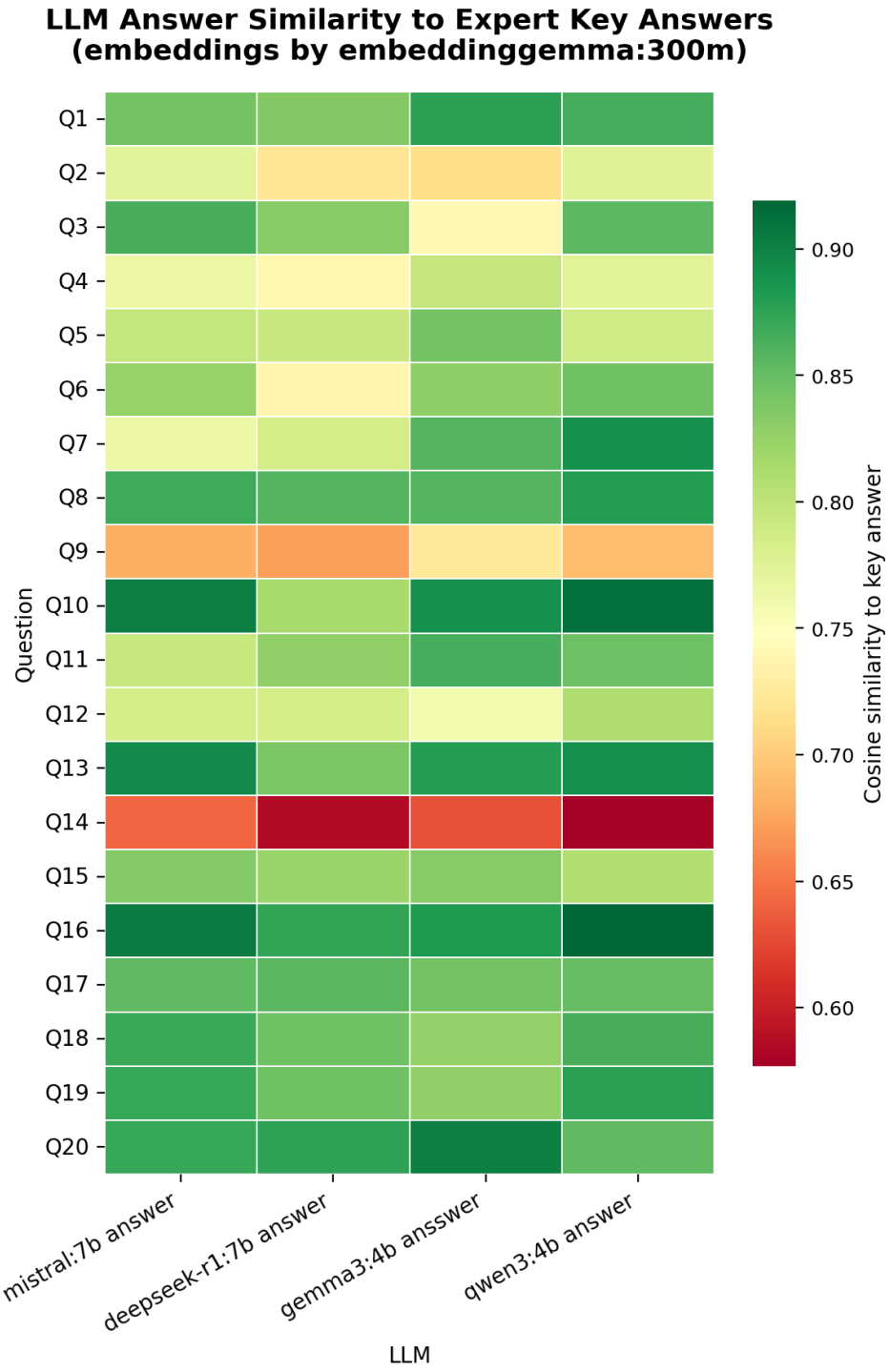


Figure 4.1. *Model selection benchmark: cosine similarity to expert answers across 20 questions and four candidate models (embeddinggemma:300m).*

qwen3:4b achieved the highest mean similarity (0.8284) and the highest number of question-level wins (8 of 20). It was selected as the generation and agent model for the subsequent phases on these grounds together with its smaller resource footprint and native tool-calling support, rather than because it was shown to be statistically superior to the other candidates.

The ranking should nevertheless be interpreted cautiously. The difference between the highest and lowest mean scores is only 0.031, which is smaller than the standard deviation observed within each model (approximately 0.07–0.08). The results therefore suggest that all four candidates have broadly similar BIM/FM vocabulary. qwen3:4b was retained because it combined the highest score with a smaller 4B resource footprint and native tool-calling support in Ollama, which was required in Phase 2.

The models also showed a similar pattern across the question set. All four performed worst on the open-ended question concerning the types of information that must be updated during building operation (0.577–0.642), while the digital-twin and COBie definitions received the highest scores (0.87–0.92). This suggests that open-ended enumeration is less predictable than definition-based answering, a pattern that also appears in Phase 3.

4.3 Construction of the retrieval-augmented agent

Phase 1 assessed the models' general knowledge. Phase 2 focused on project-specific information: the Polito Regulation and the condition records of 5,496 panels, neither of which is available in the model's training data. The resulting prototype, bim_fm_rag, was implemented as a tool-using agent rather than a single-path RAG pipeline.

4.3.1 Architectural principle: heterogeneous sources, heterogeneous handling

A central design decision was to process the two source types differently. A conventional RAG workflow would embed all content in one vector store and retrieve the most similar passages. That approach is unsuitable for the inspection register, which contains 5,496 rows with similar grade fields. A question such as how many panels were replaced requires an aggregation over the full table; retrieving the k most similar rows cannot perform a COUNT operation. Embedding the register would therefore risk producing plausible-sounding but numerically incorrect answers.

The corpus was divided according to data structure. The regulation and datasheets are treated as prose: they are chunked, embedded, and retrieved through semantic similarity. The inspection register remains structured data and is loaded into a pandas DataFrame for direct computation. The system does not use a fixed rule-based router. Instead, the model selects a tool at run time using the user's question, the system

prompt, and the tool descriptions. This run-time selection is what gives the prototype agent-like behaviour rather than the behaviour of a fixed pipeline.

Figure 4.2 summarises the hybrid architecture. During offline indexing, the prose documents are embedded once and stored in Chroma. At run time, a question from the interface is passed to the qwen3:4b agent. The agent then calls either semantic retrieval over the document collection or a direct pandas query over the register, before producing an answer together with the available source information.

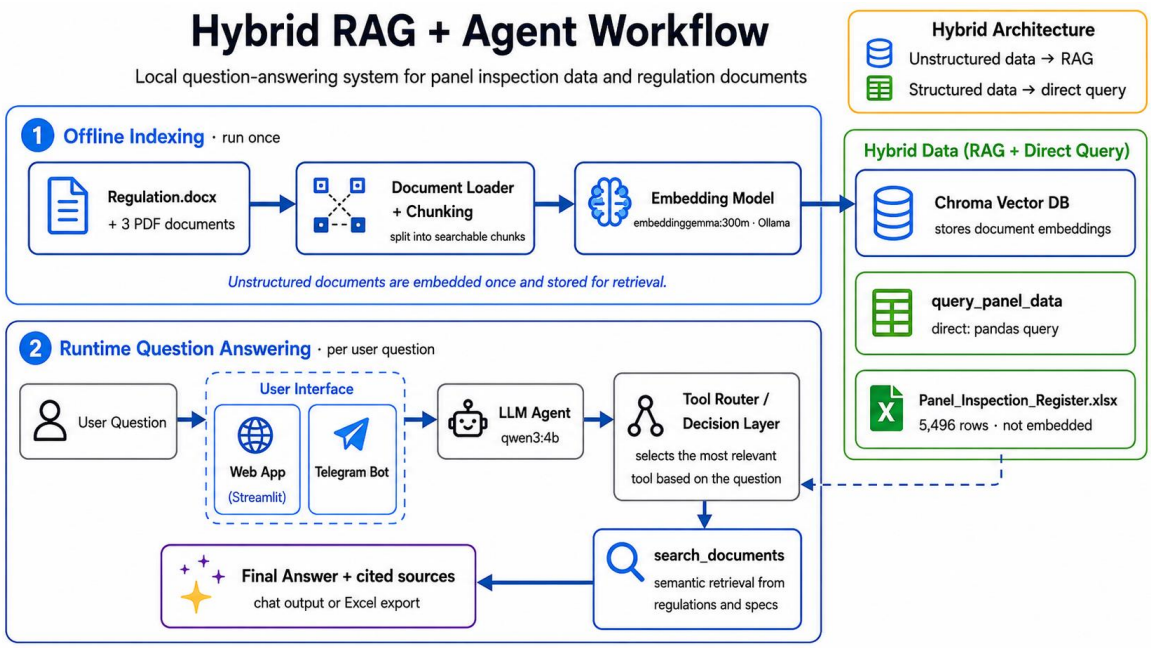


Figure 4.2. *Hybrid RAG-agent architecture, combining semantic document retrieval with direct queries over the 5,496-panel inspection register.*

4.3.2 Prose ingestion and the vector store

The vector store is created by src/ingest.py through a repeatable ingestion process.

A fixed character-based split could separate a defect grade from the action required by the regulation. The regulation is therefore divided at its 16 known section headings, listed in config.SECTION_TITLES. Each resulting chunk begins at a genuine section boundary and retains section and section_title metadata. This allows an answer to cite a specific location, such as "Polito Regulation, Section 9", which is essential when the output may inform a maintenance decision.

Two additional measures were needed. First, the extractor preserves table content because important material, including technical checklists and templates, is stored inside Word tables and would be lost by paragraph-only extraction. The document is read in order and table cells are flattened into the text stream. Second, the table of contents is excluded. In a .docx file, table-of-contents entries are returned as ordinary

text and can be mistaken for the actual section headings. The extractor recognises the ordered table-of-contents sequence, skips it, and begins section detection at the first real heading.

Long sections are further divided with a recursive character splitter using a chunk size of 800 and an overlap of 120. The splitter prefers paragraph, line, and sentence boundaries, in that order. The overlap reduces the chance that a rule spanning a boundary will be omitted from both retrieved chunks.

If the expected headings are not found, for example after a revision of the regulation or the substitution of another document, the ingestion process issues a warning and falls back to generic recursive chunking. The system continues to operate, although section-level citation becomes less precise. The three datasheets use this generic path because they do not share the regulation's known structure.

Approximately 106 chunks are produced from the four prose documents. They are embedded with embeddinggemma:300m and stored in a persistent Chroma collection. The collection is deleted and rebuilt each time the ingestion script is run, not each time a user submits a question. This prevents old and revised chunks from remaining together after a document update while avoiding unnecessary re-indexing during normal use.

4.3.3 The two tools

The agent is limited to two tools. This was a deliberate choice because a 4B model is more likely to confuse tools when several options have similar descriptions.

search documents (query) performs similarity search over the Chroma collection and returns the four highest-ranked passages. Each result includes its source document and, where available, its section number and title. The model therefore receives citation metadata directly instead of generating it. The initial value k = 4 was chosen as a practical balance between context size and retrieval coverage. It should be treated as a configurable prototype setting; a production version could evaluate dynamic k, metadata filtering, hybrid keyword-vector retrieval, or reranking on a larger validation set.

query_panel_data(code) follows a different strategy. Instead of restricting the user to a predefined list of filters, the model generates one pandas expression for the DataFrame df. The expression is executed and the result is returned as text. A fixed parser would be safer, but it could answer only question patterns anticipated during development. Expression generation supports a wider range of counts, filters, groupings, and sorting operations, including queries not defined in advance. For this prototype, that flexibility was prioritised because facility-management questions cannot be fully enumerated beforehand.

The execution environment is restricted. __builtins__ is replaced with a short allow-list containing len, sum, min, max, sorted, round, abs, and basic type constructors, while only df and pd are made available. This prevents direct imports and ordinary file access. The restriction is suitable as a hardening measure for a local, single-user research prototype, but it is not a complete security boundary. Any multi-user or externally accessible deployment should replace it with a sandboxed executor, a validated abstract syntax tree, or a constrained query language. Results are also limited to 30 rows and 3,000 characters to prevent broad queries from consuming the context window.

4.3.4 Agent orchestration and prompt engineering

The two tools are connected to qwen3:4b through LangChain's create_agent, built on LangGraph, using native tool calling. This was more reliable than a text-parsed ReAct loop for a small reasoning model whose output format may vary. Conversation state is stored in memory and separated by thread_id, which supports multi-turn interaction while keeping evaluation questions independent.

Because qwen3:4b is relatively small, the system prompt provides explicit operational guidance. It defines the assistant's role and available sources, requires answers to be grounded in tool output rather than model memory, and asks the model to cite the relevant regulation section, datasheet, or panel code. When no relevant evidence is returned, the model is instructed to state this instead of guessing. The prompt also includes the grade legend from config.py and a schema summary generated from the DataFrame at start-up, including the actual column names, data types, row count, facade values, and sample rows. Generating the schema from the loaded file prevents the prompt from becoming inconsistent with the register after a column change.

The tool docstrings also form part of the prompt. They were revised after observed failures and now explain several project-specific cases: Required Action (current) contains full sentences and should be searched with .str.contains() rather than equality; repeated grades must be counted by summing boolean comparisons across the twelve period columns with axis=1; and this calculation must not be confused with Max Grade Recorded. Canonical expressions are included for common query patterns. Later evaluation showed that the 4B model did not reliably combine both tools within one question. For mixed questions that require a register result and the regulation that explains it, a more dependable implementation would use a fixed two-stage controller that calls both tools and then asks the model to synthesise the outputs.

4.3.5 Implementation stack

All components run locally, as required in Section 4.1. Ollama hosts qwen3:4b and embeddinggemma:300m; LangChain and LangGraph manage tool orchestration and conversation state;

Chroma stores document vectors; pandas provides access to the register; and matplotlib with seaborn generate the evaluation heat maps in Section 4.4. Project data remains on the local machine, although Section 4.6 shows that this property also depends on the interface used to transport questions and answers.

4.4 Evaluation of the RAG system

4.4.1 Evaluation question set

A separate set of ten questions was prepared from the case-study documents and inspection register. Unlike the Phase 1 questions, these items require project-specific evidence and were selected to test each route through the architecture.

Table 4.4. *Evaluation question set of the RAG system*

#	Question	Path exercised
1	How many panels have been replaced so far?	Aggregation over register
2	How many curtain panels are on the facade?	Simple count
3	What is the City of Turin building regulation framework?	Regulation retrieval
4	For panel AL3G16F15, show its full inspection history 2017–2025	Single-record lookup, 12 columns
5	Which panels were replaced during Second Half 2025?	Period-specific filter
6	Which side of the building has the most failures/replacements?	Group-by aggregation
7	What do the defect grades mean in the Polito regulation?	Regulation retrieval (Section 9)
8	Which panels were replaced (grade 4) more than once, 2017–2025?	Multi-column boolean sum
9	How often should panels be inspected after 2024?	Regulation retrieval
10	In which situations should inspections be more frequent?	Regulation retrieval, conditional

One expert key answer was prepared for each of the ten questions, either by extracting the relevant regulation text or by calculating the result directly from the register. These ten reference answers are fixed ground truth, written independently of the model, and each was reused as the scoring baseline for every run of its question.

4.4.2 Repeated-run protocol

Because LLM output is stochastic, one run provides only a single sample of the system's behaviour. The ten-question set was therefore executed independently ten times, producing 100 model answers (ten questions in each of ten runs), which were scored against the ten fixed reference answers. The results

were stored in dated files (Question_Answers_run2...run10.xlsx) and consolidated in 10 Questions-10 Runs.xlsx.

Each question was placed in a separate conversation thread so that it could not reuse context from an earlier item. The runs were executed automatically, and each result was written to disk immediately. A per-question time limit prevented one stalled response from blocking the remainder of a run.

For every answer, the evaluation harness also recorded a Sources field containing the tool calls that actually occurred: retrieved passage labels and the exact pandas expression executed. This information came from the execution trace rather than from the wording of the answer. It therefore provides an independent record of system behaviour and made it possible to distinguish retrieval failures from generation failures. The same provenance information is intended for display in the interfaces discussed in Section 4.6.

4.4.3 Metric 1 — Answer fidelity

Each of the 100 answers was compared with its expert key answer using the same embeddinggemma:300m procedure and STS prefix applied in Phase 1. The resulting scores were plotted as a 10 × 10 question-by-run heat map. If a run returned no answer, it was assigned a score of 0.0 rather than being excluded, because a missing response is a genuine system outcome and an empty string has no meaningful embedding.

The mean fidelity score was 0.708 (sigma = 0.140). Figure 4.3 shows the complete matrix. The dark-red cell at Q4, run 7 represents the only empty response, while the consistently lower values for Q8 are examined in the results chapter.

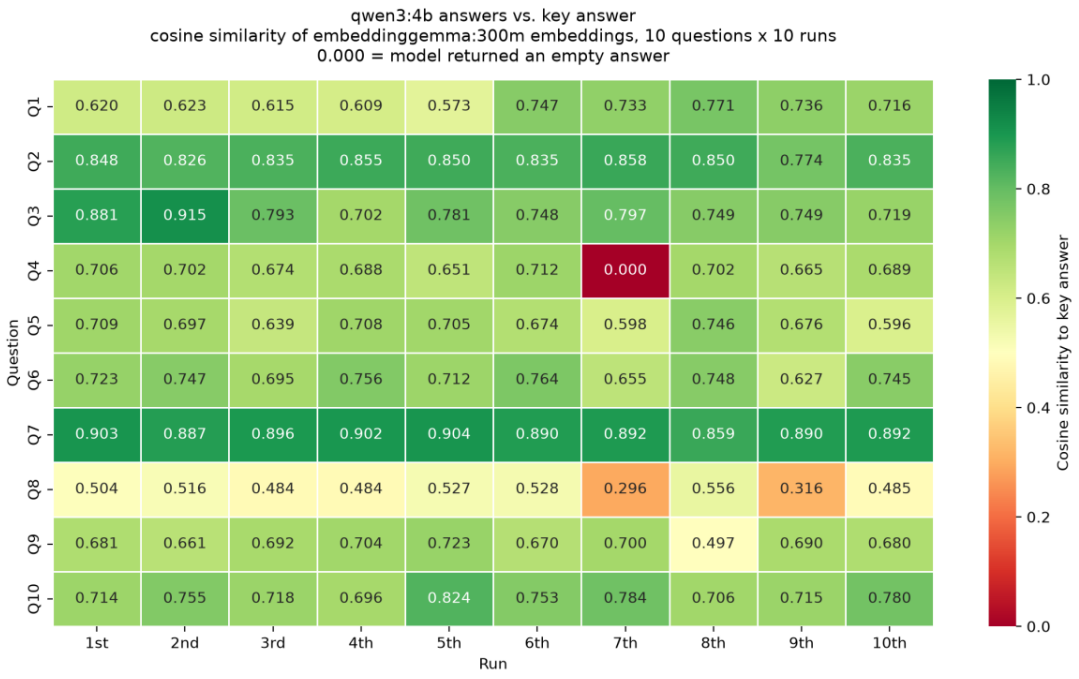


Figure 4.3. *Answer fidelity across 10 questions and 10 repeated runs, measured by cosine similarity to the expert key.*

Table 4.5. *Mean answer fidelity per question across 10 runs*

Question	Mean fidelity	Interpretation
Q7 (grade meanings)	0.891	Best — direct retrieval from Section 9
Q2 (panel count)	0.837	Simple aggregation
Q3 (Turin framework)	0.783	Regulation retrieval
Q10 (when to inspect more)	0.745	Conditional regulation reasoning
Q6 (worst facade)	0.717	Group-by aggregation
Q5 (replaced in H2 2025)	0.675	Period filter
Q1 (total replaced)	0.674	Aggregation, phrasing-sensitive
Q9 (frequency after 2024)	0.670	Regulation retrieval
Q4 (panel history)	0.619	Long tabular output
Q8 (replaced >1 time)	0.469	Worst — negative (empty-set) ground truth

The mean score for each run ranges from 0.631 to 0.733. Nine runs fall within the narrower range of 0.684–0.733, indicating that most variation is associated with particular questions rather than with entire runs. Q7 performs well throughout, whereas Q8 remains difficult. Run 7 has the lowest mean because it includes

the empty Q4 response and the lowest Q8 score (0.296). Excluding the empty cell, its mean is 0.702, which is close to the other runs. The observed instability is therefore localised to individual responses rather than reflecting a general deterioration of a complete run.

4.4.4 Metric 2 — Self-consistency

Fidelity alone has an important limitation. The key answers are short and direct, whereas the model often produces longer explanations. Cosine similarity may therefore be lower even when the longer answer is factually correct. A second metric was calculated by comparing every pair of runs for the same question. Ten runs produce 45 unique pairs per question and 450 comparisons overall. Because both texts in each comparison are full model answers, the metric focuses on whether the system gives a similar response when the same question is repeated. Table 4.6 reports this separate calculation. It uses the same answers as the fidelity analysis but a different reference point and different output files. The two metrics are related but not interchangeable, with a correlation of r = 0.82 across the ten questions. Q4 is the only case where both analyses contain a value of 0.000, caused by the empty response in run 7.

The two metrics address different practical concerns. Fidelity asks whether an answer agrees with the expert reference, while self-consistency asks whether the system responds in a stable way. Stability does not guarantee correctness, but a system whose answer changes substantially between identical requests would be difficult to use for maintenance decisions.

Averaged over the ten questions, the mean pairwise agreement was 0.879.

Table 4.6. *Self-consistency across 10 runs (45 run pairs per question)*

Question	Mean agreement	Min pair	Spread
Q7	0.951	0.903	0.096
Q2	0.931	0.877	0.115
Q6	0.919	0.850	0.136
Q9	0.918	0.788	0.196
Q3	0.896	0.778	0.222
Q1	0.884	0.787	0.188
Q5	0.884	0.816	0.145
Q10	0.884	0.808	0.177
Q8	0.764	0.480	0.521
Q4	0.757	0.000	0.984

The difference between the mean self-consistency score (0.879) and mean fidelity score (0.708) is informative. The prototype is more repeatable than accurate: it often returns a similar answer across runs, but that answer is not always fully correct. A stable systematic error is easier to investigate and correct through changes to prompts, tools, or data handling than an erratic failure with no consistent pattern.

4.4.5 Metric 3 — Manual four-class evaluation

Both automatic metrics rely on embedding similarity, which is only an indirect measure of factual correctness. Short correct answers can receive low scores, and two correct responses may differ numerically because they use different wording. Each of the 100 answers was therefore reviewed manually and assigned to one of four classes using a fixed priority rule:

1. **Error** — a blank answer, a non-answer, or a tool failure surfaced to the user;
2. **Hallucination** — an answer containing unsupported, invented detail;
3. **Wrong** — an answer that is incorrect or materially incomplete;
4. **Correct** — an answer that materially matches the key answer.

Each answer was classified according to its most serious problem. For example, an answer that is otherwise correct but introduces an unsupported detail is classified as a hallucination rather than as correct. Figure 4.4 presents the resulting matrix.

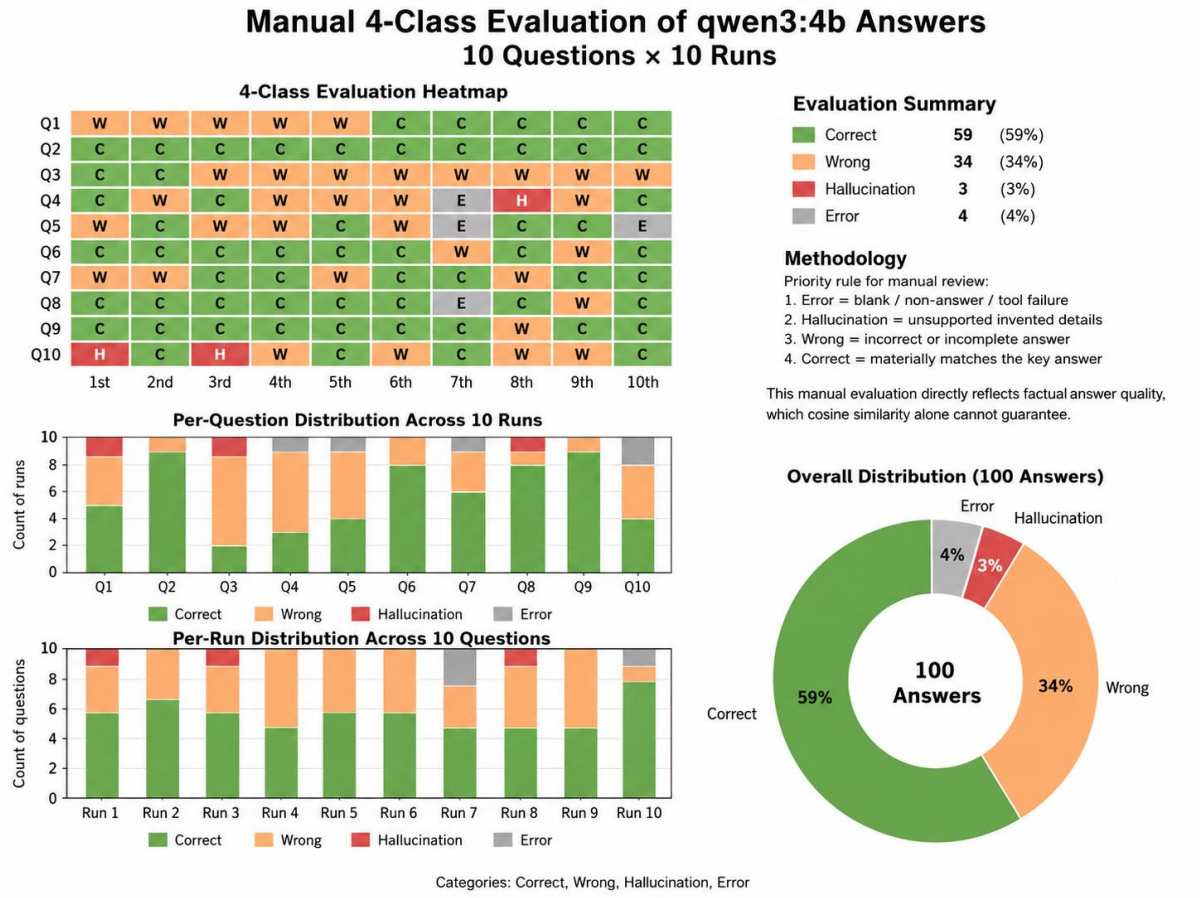


Figure 4.4. *Manual four-class evaluation of 100 answers: Correct (C), Wrong (W), Hallucination (H), and Error (E).*

Table 4.7. *Outcome of the manual four-class evaluation (100 answers)*

Class	Count	Share
Correct	59	59%
Wrong	34	34%
Hallucination	3	3%
Error	4	4%

The manual review complements the automatic metrics. Of the 100 answers, 59 were factually correct and 34 were wrong or materially incomplete. Four responses were errors, including blank outputs or visible tool failures, and three contained hallucinated details. These two higher-risk categories were uncommon but still important for a maintenance application. The heat map shows where they occurred: hallucinations were concentrated in questions where the model added context that had not been

retrieved, while the errors corresponded to empty or failed runs already visible in Figure 4.3. Manual review therefore provides the clearest direct assessment of factual answer quality.

Two features of these results are carried forward. The evaluation surfaced two systematically weaker questions (Q8, which has a negative — empty-set — ground truth, and Q4, whose single empty response in one run was an operational timeout rather than a reasoning error) and, from the recorded execution traces, showed that the agent did not combine both tools within a single question in any of the 100 recorded answers. The first point matters for interpreting the fidelity scores and the second is the basis of the tool-composition limitation discussed in Section 4.7; both are examined in detail, with the per-question and per-run breakdown, in the results chapter.

4.5 Closing the loop: BIM integration and write-back

Phases 1 to 3 implement the read path. Phase 4 addresses the write path by returning new inspection findings to the BIM model, the supporting documentation, and the retrieval corpus. This step changes the framework from a question-answering system into a two-way information workflow.

The Dynamo exchange has been implemented and tested on the Dumarey Revit model as a proof of concept: the export and import graphs run on the real model and transfer the panel code and current-period grade between the register and the Revit elements, as shown in Figures 4.7 to 4.10. The full validation battery and the complete inspection schema are specified as the production design (Section 4.5.5) rather than evaluated here. The LLM-based conversion of a free-text inspection note into a validated structured record is specified but has not yet been evaluated end to end; this boundary is discussed in Section 4.7. Figure 4.5 presents the complete loop.

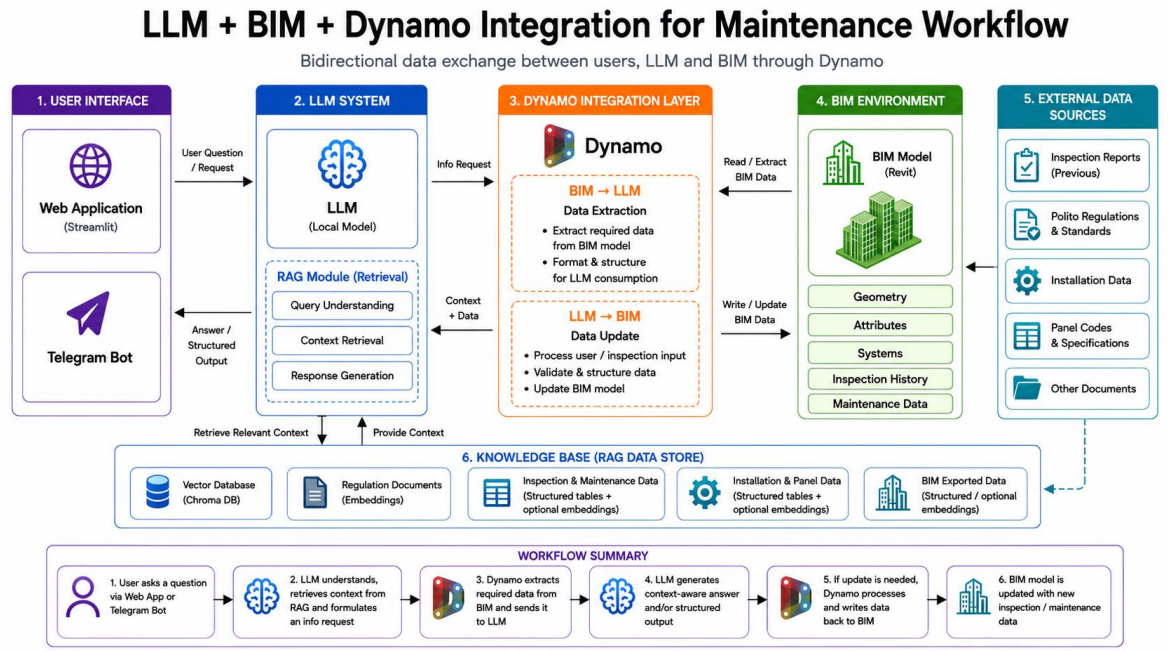


Figure 4.5. *Two-way LLM-BIM maintenance workflow linking natural-language queries, retrieval, Dynamo integration, and controlled BIM write-back.*

4.5.1 The BIM model as the authoritative condition record

At present, the spreadsheet contains the inspection history, while the BIM model mainly contains as-built geometry and type information. The panel code links the two sources, but they begin to diverge after an inspection because the spreadsheet is updated and the model is not. Over time, the model therefore represents the building as designed rather than its current condition, reducing its value during operation and maintenance.

Phase 4 changes this relationship by using the BIM model as the main record of current condition and the spreadsheet as an export of that information. The main benefit is spatial context. A grade stored on a Revit element can be visualised in three dimensions, filtered by storey or elevation, and compared with neighbouring elements. Inspection parameters can also be used in schedules, filters, and take-offs together with type, material, and level information. When exported through IFC property sets, the condition data remains available outside the original authoring application.

Dynamo provides the connection between the LLM layer and Revit. It can read and write element parameters, run Python-based validation, and exchange data with Excel without requiring a compiled add-in. Through Dynamo Player, a facility manager can run the prepared graphs without working directly in the Dynamo authoring environment.

4.5.2 The inspection parameter schema

All 5,496 panels belong to one family (Pannello sistema) in the *Curtain Panels* category, in two types (Lastra Balmoral sp. 3 cm and sp. 4 cm), so a single parameter binding covers the whole envelope.

Inspection data are added as shared parameters, bound as instance parameters to the Curtain Panels category and grouped under a custom group, *Inspection & Maintenance*. Each element of that binding was chosen for a reason. Shared rather than project parameters, because shared parameters can be reused consistently across projects and families and are required whenever the information must be exposed in tags, which is not possible with project parameters. Shared parameters also support coordinated scheduling and mapping to IFC property sets, so the same definitions can carry the condition data beyond the authoring model. Instance rather than type parameters, because condition belongs to the individual panel: binding to the type would give all 5,320 panels of the 3 cm type a single shared grade. A dedicated group, so the fields appear together in the Properties palette and can be identified programmatically as a set.

Table 4.8. *Inspection parameter schema (shared, instance-bound, Curtain Panels category)*

Parameter	Type	Source	Purpose
INSP_Panel_Code	Text	C_Codice Lastra	Join key between model, register and RAG corpus
INSP_Facade	Text	Register / code prefix	Elevation grouping (A–D)
INSP_Last_Period	Text	Inspection event	Campaign that produced the current grade
INSP_Last_Grade	Integer	Inspection event	Current defect grade, 0–4 (Regulation §9)
INSP_Max_Grade	Integer	Derived	Worst grade ever recorded at this location
INSP_Required_Action	Text	Derived from grade	Action mandated by the regulation
INSP_Last_Update	Text (ISO date)	Write-back event	When the parameter was last written
INSP_Inspector	Text	Inspection event	Attribution of the finding
INSP_Replacement_Date	Text (ISO date)	Inspection event	Set when a Grade 4 replacement is executed

The schema also defines what should remain outside Revit. The register contains twelve inspection-period columns and gains another column after each campaign. Mirroring the full-time series as Revit parameters would require continuous schema changes and would create many sparsely populated fields. Revit is not

designed to serve as a time-series database. The responsibility is therefore divided: the BIM model stores the current state and derived condition, while the register retains the complete inspection history. The model answers questions about present condition and spatial context; the register explains how that condition developed over time. The panel code connects the two.

Two fields are derived rather than observed. INSP_Required_Action is determined directly from the new grade using the Section 9 legend, so the action does not need to be typed by the inspector or inferred freely by the model. INSP_Max_Grade is updated as the maximum of the previous value and the new grade. This raises an asset-identity question: after a Grade 4 panel is replaced, the physical panel is new, but its Revit element, location, and code remain. The schema retains the historical maximum and records INSP_Replacement_Date separately. This choice treats the maximum grade as a property of the location's history, while the replacement date preserves the age of the current panel. It also remains consistent with the existing register. The distinction has not yet affected the data because none of the 150 panels that reached Grade 4 has done so twice, but the rule must be defined before a recurrence occurs.

4.5.3 The panel code as join key

The loop depends on C_Codice Lastra being a reliable primary key, which was verified against the register: all 5,496 codes are unique, so a lookup resolves to exactly one element. Two further properties were established, and both fed into the write-back design.

The code is not fixed-width. Most codes are 9 characters (5,320 panels), but 168 are 11 characters and 8 are 10. Matching logic must therefore compare whole strings, since any routine that slices the code by character index will silently mismatch those 176 panels.

The first character redundantly encodes the elevation. Codes beginning A belong to facade A (2,398 panels), B to facade B (1,815), C to facade C (629) and D to facade D (654), without exception. That redundancy is used as a validation rule: a record whose code prefix disagrees with its stated facade is almost certainly a transcription error and is rejected before it reaches the model.

4.5.4 Inspection capture and validation

An inspection campaign produces a new dataset, and two capture routes are supported.

The primary route is a structured field worksheet. Before a campaign, the export graph writes an Excel sheet pre-populated with every panel code, facade, level and current grade, leaving the new grade and notes blank. The purpose of pre-filling is that the inspector never types a panel code, which removes the largest single source of write-back failure. The returned file is the new inspection Excel.

The secondary route supports natural-language reporting. An inspector may submit a note such as "panel AL3G16F15, visible cracking around the lower anchor, looks like grade 3" through the interface. The LLM converts the note into the same structured record used by the worksheet. Grade definitions are retrieved from Section 9 through search_documents so that classification is grounded in the regulation rather than based only on the model's judgement. The worksheet is more appropriate for full campaigns involving thousands of panels, while natural-language capture is intended for individual findings between campaigns.

Every record is validated before it can update the model. An incorrect answer may be noticed and corrected, but an incorrect parameter value can remain in the authoritative record and affect later decisions. Validation is therefore completed before a Revit transaction begins, and rejected rows are written to an exception report rather than being silently discarded.

Table 4.9. *Validation rules applied to every inspection record*

Rule	Basis	Action on failure
Panel code exists in the model	Model element lookup	Reject; report unmatched code
Panel code is unique in the submitted file	Key integrity	Reject duplicate rows
Grade is an integer in 0–4	Regulation §9	Reject; out-of-range grade
Required action matches the grade	Regulation §9 legend	Correct to the mandated action, flag the discrepancy
Code prefix agrees with stated facade	Verified redundancy (§3.6.3)	Reject; probable transcription error
Inspection period is a recognised campaign	Register schema	Reject; unknown period
Grade 4 carries a replacement date	Regulation maintenance rules	Flag as incomplete

4.5.5 The Dynamo graphs

Two graphs implement the exchange, both runnable from Dynamo Player so that the manager selects a file and presses run. Dynamo acts as the bidirectional translator between the model and the LLM layer, extracting panel data in one direction and writing validated updates in the other; Figure 4.6 sets out those two roles.

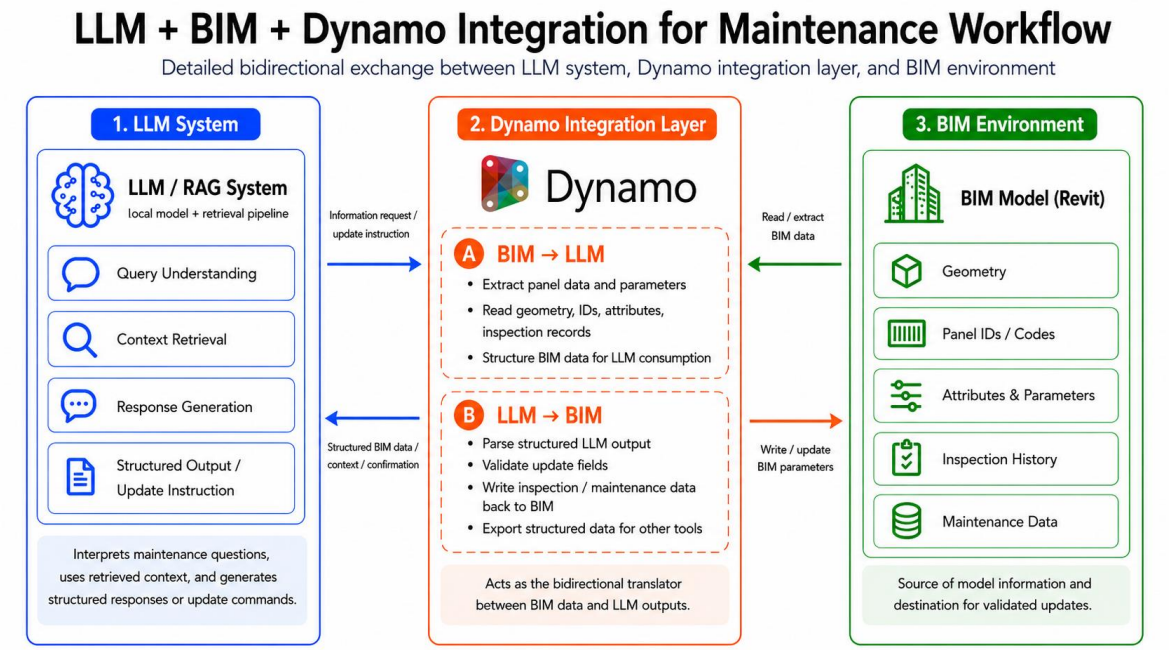


Figure 4.6. *Dynamo integration layer for BIM-to-LLM data extraction and validated LLM-to-BIM write-back.*

It is useful to distinguish here between what has been implemented and demonstrated on the real model and what is designed as the complete production workflow. The export graph and the write-back graph were both built and run in Dynamo against the Dumarey Revit model, and the round trip demonstrated in Figures 4.7 to 4.10 transfers the panel code and the current-period grade between the register and the corresponding Revit elements. This is the implemented proof of concept. The remaining components described in this section — the full validation battery of Table 4.9, the dictionary-based lookup, the unmatched-code exception report, and the complete INSP_ schema of Table 4.8 — are specified and, where noted, partially prototyped, but they are presented here as the proposed production design rather than as fully evaluated features. Sentences describing them are therefore phrased as design intent, and Section 4.7 identifies which parts remain to be measured.

The export graph reads the model and produces either the inspection worksheet or the regenerated register. Categories feeding All Elements of Category collects the 5,496 panel instances; Element.GetParameterValueByName extracts the code, facade, level, type and inspection parameters as parallel lists; List.Transpose restructures them into row-per-panel form; and Data.ExportToExcel writes the sheet with a header row matching the register's column names exactly, so the export is drop-in compatible with the RAG read path.

The write-back graph reads the completed inspection file and updates the model. File Path and File.FromPath feed Data.ImportExcel with readAsStrings set to true, which is not optional: read as typed

data, Excel coerces mixed-format identifiers and numeric-looking fields, corrupting codes and turning grades into floating-point values. Conversion to integer happens explicitly afterwards, where it can be checked. In the proposed production design, the imported rows then pass through the validation rules of Table 4.9, carried out in a Python Script node so the logic stays legible and testable rather than being spread across dozens of graph nodes. In the demonstrated proof of concept, the transferred fields are the panel code and the current-period grade, so the full validation battery is specified for the complete implementation rather than executed in the graph shown in Figure 4.8.

The model elements are collected in the same way as in the export graph. In the proposed production design a dictionary is created with Dictionary.ByKeysValues, using INSP_Panel_Code as the key and the Revit element as the value. Each incoming row can therefore find its target through one Dictionary.ValueAtKey lookup. This is more efficient than comparing every spreadsheet row with every model element through nested lists, which would require roughly 30 million comparisons for 5,496 records. Codes found in the file but not in the model would be identified with List.SetDifference and written to an unmatched-code report. Such a code may indicate a typing error, a removed panel, or a difference between the register and the current model. In each case, the record requires review and should not be ignored.

Element.SetParameterByName writes the values onto the matching elements. In the graph demonstrated these are the panel code and the current-period grade; in the proposed production design they are the full set of validated fields defined in Table 4.8. Parameter names must match exactly, since this node fails without raising when given a name the element does not carry, which is a further argument for the fixed schema of Table 4.8. Writing only the elements whose values actually changed is part of the same design, keeping the Revit transaction, undo stack and journal proportional to the size of the change rather than to the size of the model.

Two execution properties were included. First, the graph is idempotent because it assigns absolute values rather than incrementing existing values. Running the same file again therefore produces the same model state. Second, when the project uses Revit worksharing, the affected elements must be borrowed, and the local model synchronised with the central file. Updating only a local copy would recreate the separation between the BIM model and the inspection record that this phase is intended to remove.

Both graphs were built and run against the real Dumarey model. Figure 4.7 shows the export graph, in which the *Curtain Panels* category is collected, filtered to the elements carrying a C_Codice Lastra, its parameters read, and the result written to Excel. Figure 4.8 shows the write-back graph as implemented

for the proof of concept, in which the completed workbook is read with Data.ImportExcel (with readAsStrings enabled, visible in the node), transposed, split into the panel-code and current-period-grade columns, and written back onto the matching elements with Element.SetParameterByName. The dictionary lookup, the full validation node, and the unmatched-code report described above belong to the proposed production graph and are not shown in this demonstration.

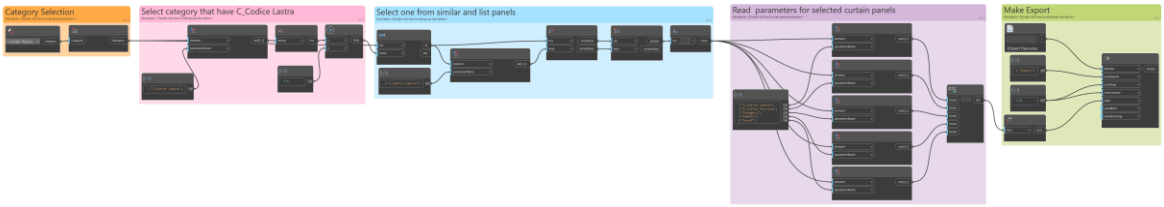


Figure 4.7. *Dynamo export graph for BIM-to-Excel panel and inspection-data extraction.*

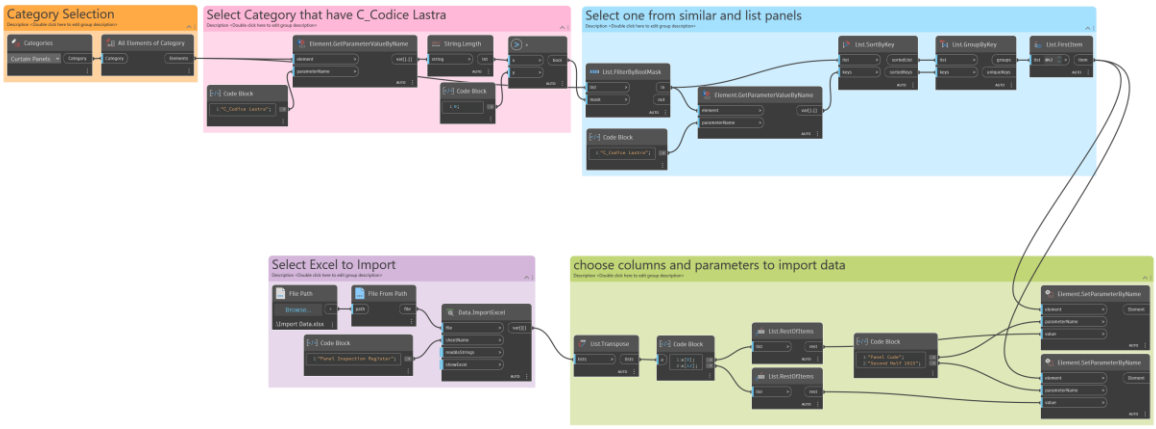


Figure 4.8. *Dynamo write-back graph for importing inspection data from Excel and updating the matching Revit panels.*

Figures 4.9 and 4.10 show the result in Revit. Figure 4.9 presents the Panel Schedule, where each panel code is listed together with its imported grade for the current period. Figure 4.10 shows the properties of one selected panel after the inspection fields have been populated. Together, these figures confirm the round trip: a value entered in the inspection workbook is written to the correct Revit element and becomes available as a queryable parameter.

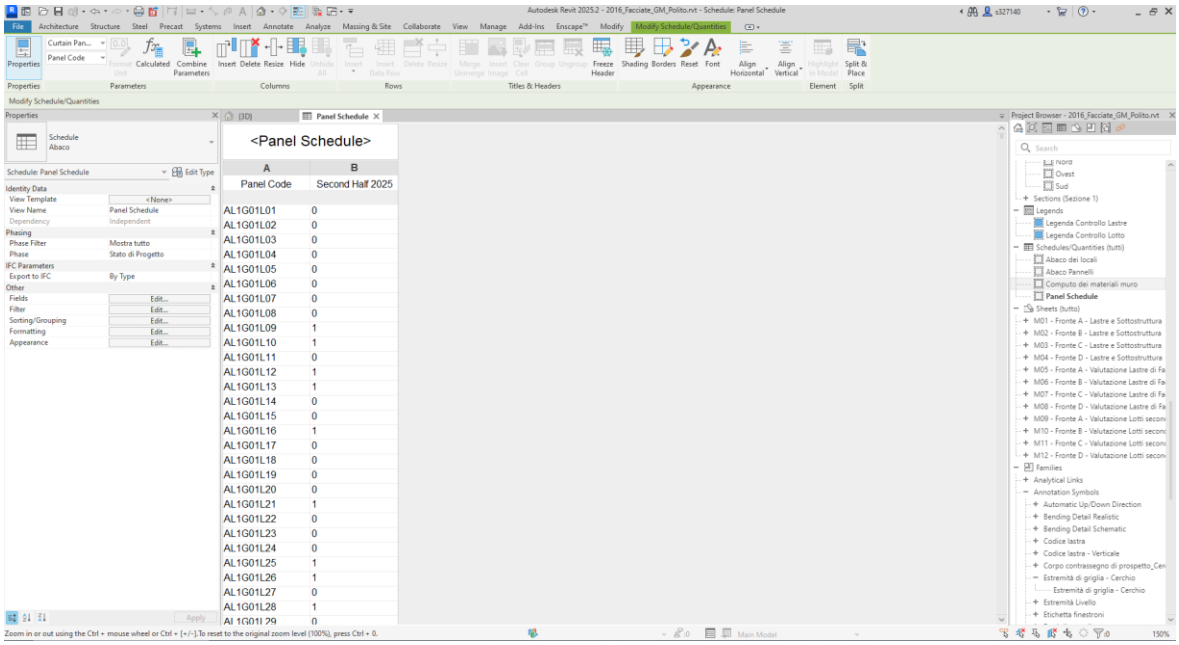


Figure 4.9. *Revit Panel Schedule after write-back, showing panel codes and the imported current inspection grade.*

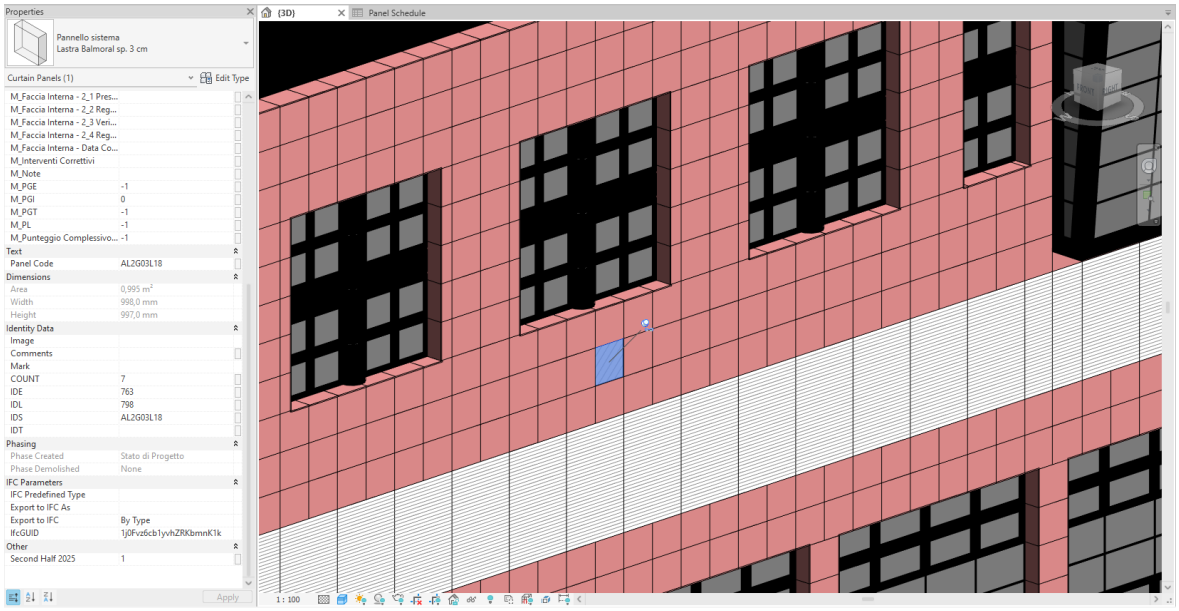


Figure 4.10. *Curtain Panel properties after write-back, showing the panel code and imported current-period inspection grade.*

4.5.6 Propagation to the documentation and the RAG corpus

Updating the BIM model is only one part of the loop. The new information must also become available to the retrieval layer so that the facility manager can query it. Two update channels are used, with different mechanisms and response times.

Table 4.10. *Propagation of an update to the retrieval layer*

Artefact	Mechanism	Re-embedding required	Latency
Panel Inspection Register (.xlsx)	Regenerated from the model by the export graph	No	Immediate on next query
Regulation, datasheets, inspection reports (prose)	Re-run src/ingest.py	Yes	Minutes
Agent schema description	Regenerated from the DataFrame at start-up	No	On next start-up

This update process benefits directly from the architectural decision in Section 4.3.1. The register is not embedded and is read as structured data, so an updated inspection becomes available after the export graph regenerates the file. There is no need to rebuild a vector index for all 5,496 rows, and old and current grades cannot coexist as separate embedded passages. The prose documents change less frequently and can therefore absorb the occasional re-ingestion cost.

The prompt schema is also generated from the loaded DataFrame rather than being hard-coded. When a new inspection-period column is added, it appears in the schema description the next time the agent starts, allowing the model to query a period that did not exist when the original prompt was written.

One additional step is required in a long-running session. panel_data.py caches the register in a module-level variable and normally loads it once per process. After the export graph rewrites the file, a running session would continue to use the old DataFrame unless it is refreshed. The loader already provides a force_reload argument. The interface should call it after write-back, either through a file-change check or an explicit refresh action. Without this reload, the loop would be updated on disk but not in memory.

4.5.7 Interoperability and governance

The case study uses Revit and Dynamo, but the inspection data is not intended to remain dependent on those applications. Shared parameters can be mapped to IFC property sets so that condition information is available in other IFC-compatible systems, including CAFM and CMMS platforms. The same mapping can support COBie deliverables. The export graph can also generate operational outputs directly: filtering INSP_Required_Action can identify panels requiring detailed inspection or replacement, while filters based on grade, level, and elevation can support maintenance schedules.

The evaluation results also define an important governance requirement for the write path. The agent does not reliably combine tools, and a meaningful proportion of its answers are imperfect. Its structured output should therefore never be written directly on the authoritative record. The LLM proposes the

record, an inspector or facility manager reviews and approves it, and only then does the Dynamo graph commit the update. INSP_Inspector and INSP_Last_Update record who approved the change and when. With a mean fidelity of 0.708, the system should be used as decision support, not as an unsupervised author of safety records.

4.6 User interface: presenting the answer to the user

4.6.1 Role of the presentation layer

Sections 4.3 and 4.4 explain how the system produces an answer and how its performance was evaluated. This section considers how that answer is delivered to facility managers and inspectors. During development, the agent was accessed through a command-line chat and the evaluation harness. Neither is suitable for routine use. The interface is therefore part of the methodology because it affects whether users can understand, verify, and act on the output.

The agent provides one text input and returns one response payload, which keeps the presentation layer separate from the reasoning layer. The interface can therefore be changed without modifying the agent itself. This separation allows the two options below to be compared with the same underlying functionality.

4.6.2 What must be displayed

A response from this system includes more than the model's written answer. It also contains provenance information: the regulation sections and datasheets retrieved, together with the exact pandas expression executed against the register.

This provenance should be visible to the user. The mean fidelity score of 0.708 indicates that a substantial proportion of responses are imperfect, and the score alone cannot identify whether a problem occurred during retrieval, tool execution, or answer generation. Showing the query and retrieved sources allows a facility manager to check how a result was obtained. At the current level of performance, provenance is a functional requirement rather than an optional interface feature.

The evaluation leads to three further interface requirements. First, tables are necessary for long panel histories and panel lists, which are difficult to read as continuous prose. Second, citations should link back to the relevant regulation section whenever metadata are available. Third, failure states must be explicit. The interface should distinguish between a missing response and a valid answer stating that no records meet the condition, because these cases occurred separately in Q4 and Q8.

4.6.3 Option A — Web application

Option A is a self-hosted web application running on the same machine as the agent. The interface was developed to the design stage only and was not evaluated as a working system; Figures 4.11 to 4.14 are therefore design mock-ups illustrating the main views: a chat interface with table rendering, a project view, a document view for the ingested corpus, and a BIM model view that connects a selected panel to its code, category, latest inspection, and current grade.

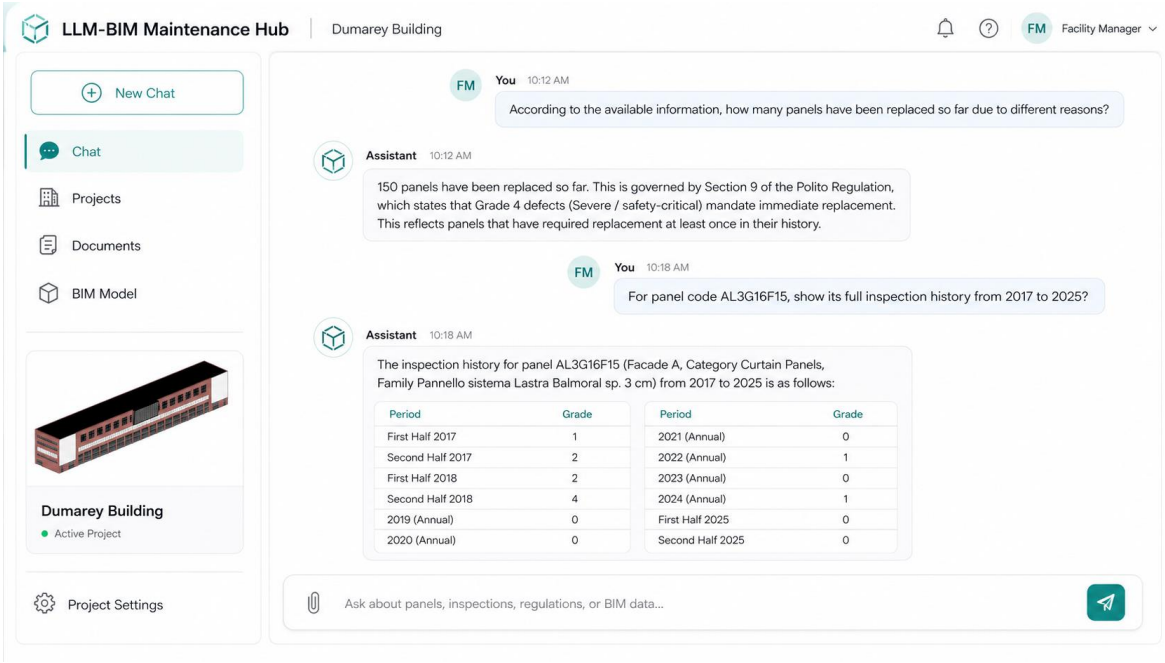


Figure 4.11. *Conceptual chat-interface mock-up showing natural-language queries, cited answers, and tabular inspection-history output.*

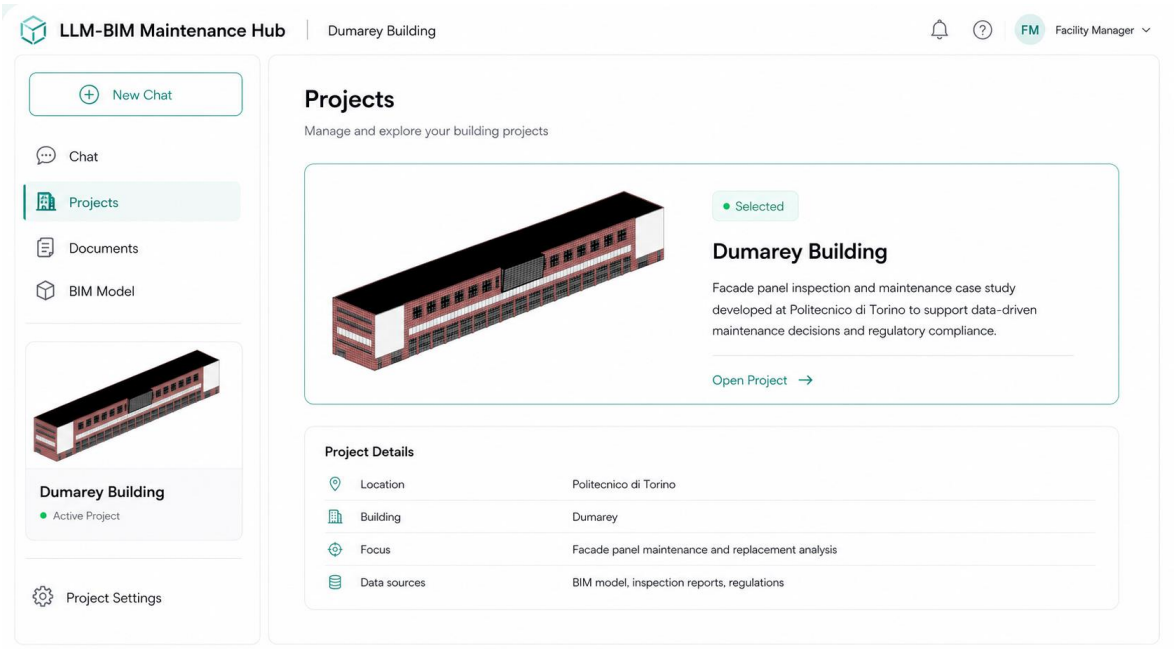


Figure 4.12. *Interface mock-up, project view: the Dumarey building, project focus, and available data sources.*

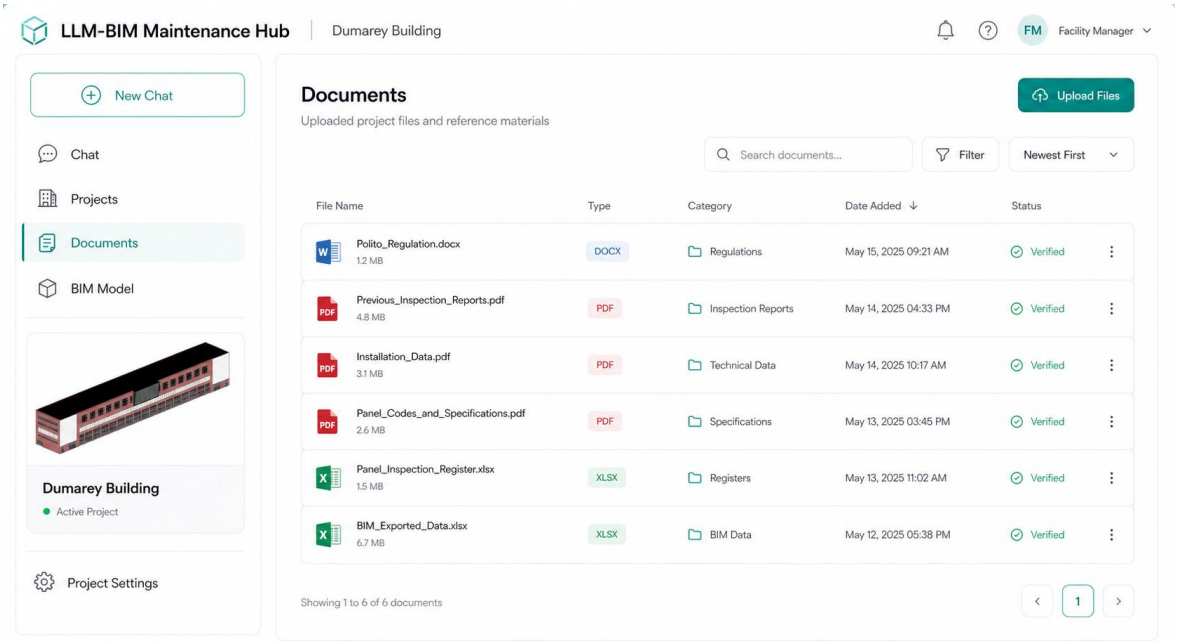


Figure 4.13. *Interface mock-up, documents view: ingested regulations, reports, specifications, and inspection-register sources.*

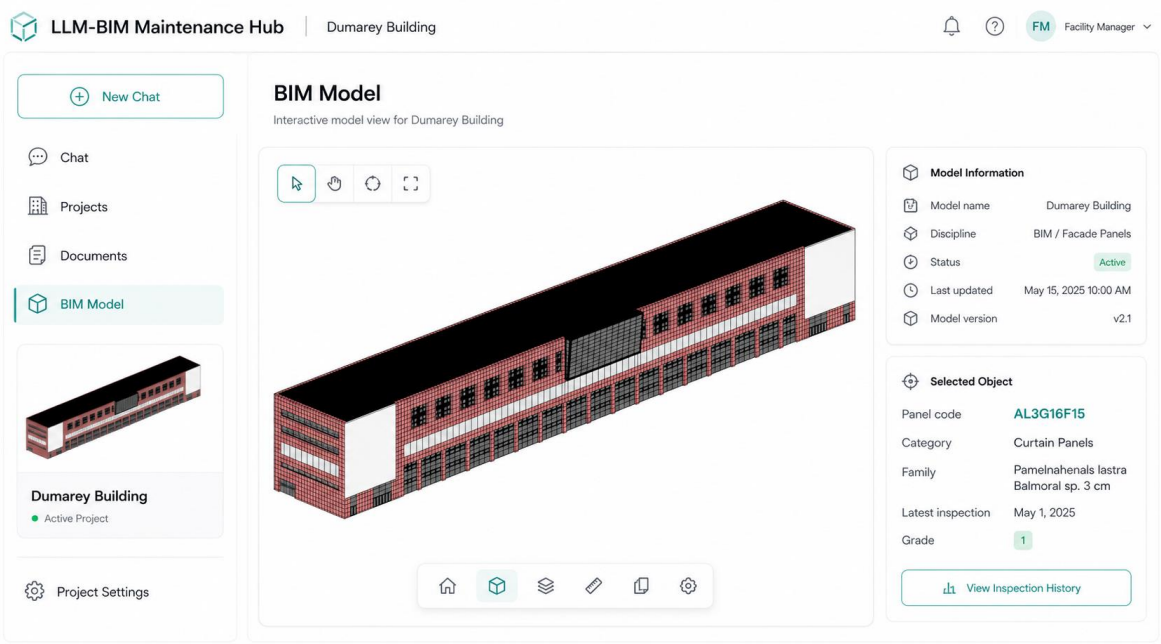


Figure 4.14. *Interface mock-up, model view: a selected panel linked to its code, category, latest inspection, and current grade.*

The main advantage of the web application is control over presentation. Query results can be displayed as sortable HTML tables, provenance can be placed in a collapsible panel, citations can link to regulation sections, and histories or evaluation results can be plotted. This is particularly valuable when an answer contains a long list of panel codes or several inspection periods.

A locally hosted web application also preserves the data-locality requirement from Section 4.1 for both inference and transport. Access control, session management, and audit logging can remain under institutional control. This would allow the human-confirmation step in Section 4.5.7 to be implemented as a traceable approval linked to a named user.

The disadvantages are mainly practical. A web application requires more development and deployment effort, including front-end, back-end, and authentication work. It also assumes access to a browser, which is suitable for a facility manager working at a desk but less convenient for an inspector standing at the facade.

4.6.4 Option B — Telegram bot

The second option delivers answers through a Telegram bot: the user sends a question as a chat message and receives the answer as a reply.

The Telegram bot offers fast and familiar access. The user does not need to install a separate application, and the chat format handles delayed local inference more naturally than a web page waiting for a synchronous response. The platform also provides an existing identity and messaging layer.

Telegram is also well suited to individual field observations. An inspector can report a finding from a phone while standing near the panel, and inline confirmation buttons could support the human-approval step before write-back.

The limitations are mainly related to presentation. Telegram messages provide limited table formatting, so long histories and panel lists would need to be sent as monospaced text or attachments. Message-length limits may also truncate the types of results produced by Q4 and Q5. Provenance would either make every reply longer or be placed behind an additional command, where it may be overlooked.

Telegram also weakens the confidentiality guarantee. The LLM can still run locally, but questions and answers, including panel codes and defect grades, are transmitted through third-party infrastructure outside institutional control. This does not automatically exclude the option, but its use would require an explicit institutional decision and appropriate data-governance approval.

4.6.5 Comparison and selection

Table 4.11. *Comparison of the two interface options*

Criterion	Web application	Telegram bot
Tabular result rendering	Full (HTML tables)	Limited (text / attachment)
Provenance display	Collapsible panel, always available	Verbose inline or hidden behind a command
Citation traceability	Hyperlink to regulation section	Plain text reference
Handling of inference latency	Requires explicit async handling	Native to the medium
Field use by inspectors	Poor (desk-bound)	Strong (phone, in situ)
Phase 4 write confirmation	Auditable, attributable approval	Inline buttons, platform-bound identity
Data locality (Section 4.1)	Fully preserved	Transport leaves institutional control
Development and deployment effort	High	Low

The two interfaces serve different parts of the same workflow. The web application is the stronger primary interface for the read path because it can display tables, sources, and aggregate results while preserving local data transport. Telegram is better suited to short field observations in the Phase 4 write path, where

mobile access is more important than rich presentation. Its use would remain subject to institutional approval because of the transport of project data through an external platform.

A staged implementation follows from these requirements. The web application should be developed first because table rendering and provenance display are necessary for a system with a mean fidelity of 0.708. A Telegram interface could then be added for field capture after the confirmation and data-governance procedures have been defined.

4.7 Methodological limitations

Six methodological limitations remain.

1. Embedding similarity is an imperfect proxy for correctness. A factually correct response may receive a lower score than a fluent but incorrect response that resembles the wording of the key answer. Q8 demonstrates this problem for short statements of absence. The manual four-class review in Section 4.4.5 reduces this limitation, but it was conducted as a single-reviewer assessment, and no inter-rater agreement was measured. A second independent reviewer and a documented agreement measure would strengthen the validity of the manual classification.

2. The agent did not compose tools. Across all 100 recorded answers, it never called both tools for the same question. Mixed questions that require a data result together with the regulation that explains it are therefore not yet reliable. This can be addressed without abandoning local execution by using a deterministic two-stage controller that calls both tools, or by repeating the evaluation with a larger local model.

3. Phase 3 evaluated only qwen3:4b. The Phase 1 difference between the four models was only 0.031 in mean similarity, which does not establish that qwen3:4b is also the best model for instruction following and tool use within the complete retrieval architecture. Repeating Phase 3 with mistral:7b would provide a direct comparison.

4. The register represents one facade typology in one building. All 5,496 records concern curtain panels installed in 2016. The results have not yet been tested on portfolios with different component types, construction periods, or multiple buildings.

5. The pandas execution environment is restricted but not fully sandboxed. The current allow-list is appropriate for local single-user research use, but it is not sufficient for an interface exposed to multiple or untrusted users. A production deployment would require a sandboxed executor, validated query representation, or constrained query language.

6. The write path has been implemented in Dynamo but has not been evaluated as a complete end-to-end process. The export and import graphs run on the real model, as shown in Figures 4.7 to 4.10, but the LLM step that converts a free-text observation into a validated structured record remains unmeasured. A suitable evaluation would use a set of inspection descriptions with known structured outputs, repeat each case, and score the individual fields by exact match. Until this is completed, human confirmation before write-back is essential.

4.8 Summary

The methodology progresses from local-model selection to system construction and empirical evaluation, followed by BIM integration to complete the information loop. Four contributions are central to the approach.

First, the architecture reflects the structure of the source data. Prose is retrieved semantically, while structured records are queried computationally. This avoids using vector retrieval for table-wide calculations over the 5,496-row register.

Second, the evaluation separates answer accuracy from answer stability. Across 100 responses, the prototype achieved a mean fidelity of 0.708 against expert key answers and a mean self-consistency of 0.879 across repeated runs. The difference shows that the system is more repeatable than accurate and indicates that correction should focus on systematic errors in prompts, tools, and data handling.

Third, the execution trace records the tools called and the queries executed independently of the model's own explanation. This made it possible to identify the empty Q4 response as an operational failure and the absence of tool composition across all 100 recorded answers as a model-capacity limitation. The same record should be exposed by the user interface so that users can verify how an answer was obtained.

Fourth, the workflow is designed to close the information loop between inspection records and the BIM model. A Dynamo round trip keyed by the verified panel code was implemented and demonstrated on the real model, transferring the panel code and current-period grade between the register and the corresponding Revit elements and exporting the result back to the register. The complete loop — in which field findings are validated against the regulation and the full inspection schema is written to instance parameters — is specified as the production design, with the parts that remain to be evaluated identified in Section 4.7. Because the register is queried directly rather than embedded, the new data become available without re-embedding. The BIM model can therefore function as an up-to-date, spatially navigable record of facade condition while remaining interoperable through IFC property sets.

5. Results and Discussion

This chapter reports and interprets the empirical results of the framework described in Chapter 4. It presents the outcome of the three completed evaluation phases and draws them together into an assessment of what the prototype can and cannot reliably do. Phase 1 selected the generation model by comparing four locally executable Large Language Models (LLMs) against expert reference answers. Phase 3 then evaluated the selected model inside the complete retrieval-augmented architecture, using three complementary measures: answer fidelity to a fixed expert key, self-consistency across repeated runs, and a manual four-class review of factual correctness. The results are read here through the same figures produced in Chapter 4, so that the discussion refers directly to the evidence rather than to summary statistics alone.

A single analytical principle organises the discussion. Each result is presented as a matrix of questions against either models (Phase 1) or repeated runs (Phase 3), which makes it possible to separate three very different kinds of weakness. A weak row — a question that scores poorly across every model or every run — indicates a property of the question itself or of the domain, not a momentary failure. A weak single cell, surrounded by strong neighbours, indicates a localised operational or stochastic failure in one run. A weak column would indicate that an entire run degraded systemically. This distinction is used throughout the chapter: it explains why questions 9 and 14 are weak for every model in Phase 1, why question 8 is weak in every run of Phase 3, and why the empty cell at question 4, run 7 must be interpreted differently from either of these.

5.1 Selection of the local model

Table 5.1 summarises the benchmark of the four candidate models over the twenty BIM and facility-management questions, each answer scored by cosine similarity to an expert reference answer using the embeddinggemma:300m model. Figure 5.1 shows the full 20×4 matrix of scores from which the table is derived.

Table 5.1. *Model selection benchmark results (cosine similarity to expert reference answers, 20 questions)*

Model	Mean	Std. dev.	Min	Max	Wins (of 20)
qwen3:4b	0.8284	0.0805	0.5769	0.9192	8
mistral:7b	0.8203	0.0708	0.6424	0.9059	5
gemma3:4b	0.8189	0.0712	0.6293	0.9027	6
deepseek-r1:7b	0.7973	0.0739	0.5858	0.8764	1

qwen3:4b obtained the highest mean similarity (0.8284) and the largest number of question-level wins (8 of 20), and it was selected as the generation and agent model for the remaining phases. The selection rests on three grounds taken together: the highest mean score, a compact 4-billion-parameter footprint suitable for the CPU-only local runtime, and native tool-calling support in Ollama, which Phase 2 required. It does not rest on a claim of statistical superiority. The difference between the best and worst mean scores is only 0.031, which is smaller than the standard deviation observed within each model, approximately 0.07 to 0.08. In practical terms, the four candidates carry broadly similar general knowledge of the field, and the ranking should be read as a reasonable basis for choosing one model rather than as proof that the others are inferior.

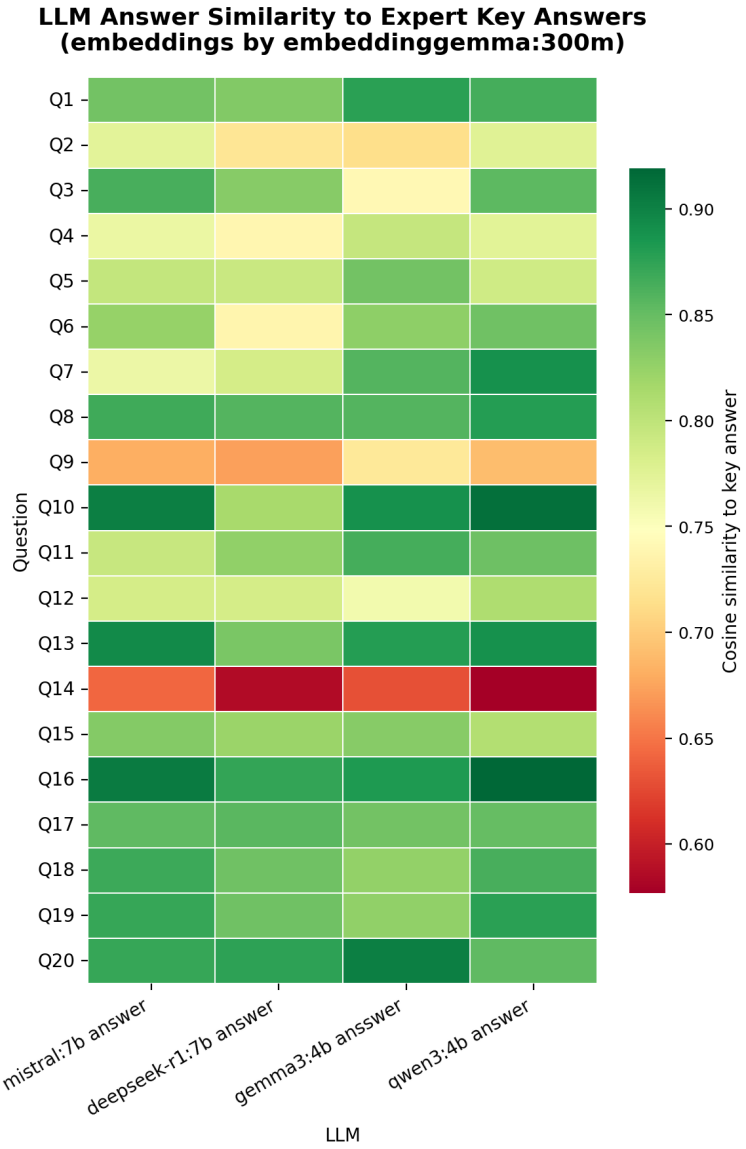


Figure 5.1. *Model selection benchmark: cosine similarity to expert answers across 20 questions and four candidate models.*

5.1.1 Why questions 9 and 14 are weak for every model

The most informative feature of Figure 5.1 is not the small gap between the models but the strong pattern shared by all of them. The heat map is organised almost entirely by row rather than by column: the same questions are easy for every model, and the same questions are hard for every model. Two rows stand out as weak across the entire matrix. Question 14, “What types of data need to be updated in a BIM model during building operations?”, is the single weakest question and forms each model’s minimum score — from 0.577 for qwen3:4b to 0.642 for mistral:7b. Question 9, “What role does a facility manager play in

defining BIM requirements?", forms a second, slightly less severe band in the range of roughly 0.65 to 0.72. No model escapes either row.

Because the weakness is uniform across four independently trained models, it cannot be attributed to a defect in any one of them. It is a property of the questions and of the domain they probe, and it has two related causes. The first is the form of the question. Both items ask for an open-ended enumeration — a list of the data that change during operation, or a description of a role defined across several documents — rather than for a single canonical definition. There is no short, agreed answer against which a model response can align tightly. Each model produces a reasonable but differently framed and differently ordered list, and cosine similarity to one fixed reference answer necessarily falls, even when the content is defensible. The contrast with the strongest rows makes this clear: Q10 (COBie) and Q16 (digital twin), which received the highest scores of 0.87 to 0.92, are closed definitional questions with a well-established, compact answer that every model reproduces in almost the same words.

The second cause is specific to the subject of these two questions, and it connects the benchmark directly to the motivation of this thesis. Both weak questions concern the operation and maintenance phase: the information that must be maintained as a building is used, and the facility manager's role in specifying that information. This is precisely the area that the literature review identified as the least mature use of BIM, where models built for design coordination are weakest and where structured asset information is most often missing. The general-purpose training data behind these LLMs reflects that imbalance: definitional and design-stage knowledge is well represented, whereas the operational, requirements-defining, and lifecycle-maintenance material is sparser and less standardised. The models are therefore least confident exactly where the maintenance workflow needs them most.

This finding is one of the strongest arguments in the thesis for the retrieval-augmented design that follows. If the models' unaided knowledge is weakest on operational and maintenance questions, then a maintenance assistant cannot depend on that knowledge alone. It must ground its answers in project-specific documents — the regulation, the datasheets, and the inspection register — through retrieval and computation. Phase 1 thus does more than choose a model; it demonstrates the gap that Phases 2 and 3 are built to close.

5.2 Answer fidelity

Phase 3 evaluated the selected model inside the full retrieval-augmented agent, using the ten case-study questions of Table 4.4, each executed independently ten times to produce 100 answers. The first measure,

answer fidelity, compares each answer with its fixed expert key by cosine similarity. Figure 5.2 presents the complete 10×10 matrix of question against run, and Table 5.2 gives the mean fidelity of each question across its ten runs.

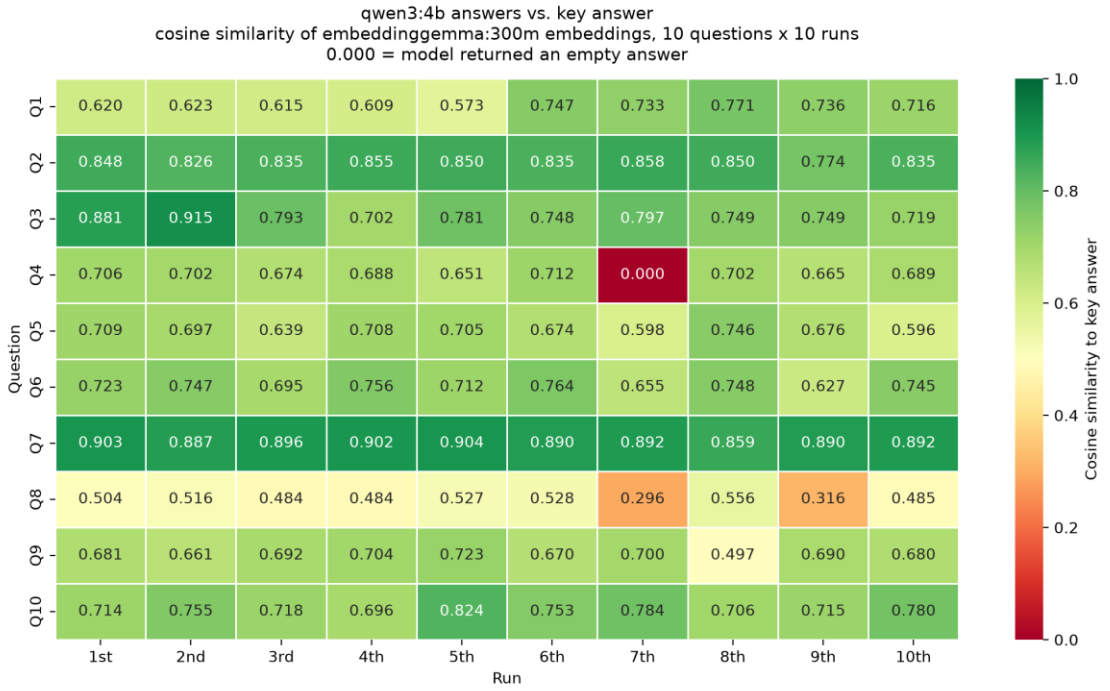


Figure 5.2. *Answer fidelity across 100 responses (10 questions x 10 runs), measured against expert key answers.*

The mean fidelity over all 100 answers was 0.708, with a standard deviation of 0.140. The per-run means fall in a narrow band from 0.631 to 0.733, and nine of the ten runs lie within the even tighter range of 0.684 to 0.733. This is an important structural result in its own right: the variation in the matrix is organised by row, not by column. Most of the difference in scores is attributable to particular questions, while whole runs behave similarly to one another. There is no weak column, and therefore no evidence that the system as a whole degrades or drifts from one execution to the next.

Table 5.2. *Mean answer fidelity per question across 10 runs, ordered from best to worst*

Question	Mean fidelity	Path exercised	Interpretation
Q7 – grade meanings	0.891	Regulation (§9)	Best – direct definitional retrieval
Q2 – panel count	0.837	Simple count	Stable aggregation
Q3 – Turin framework	0.783	Regulation retrieval	High score, but see §5.4
Q10 – when to inspect more	0.745	Conditional regulation	Conditional reasoning
Q6 – worst facade	0.717	Group-by aggregation	Grouped aggregation
Q5 – replaced in H2 2025	0.675	Period filter	Filter, phrasing-sensitive

Question	Mean fidelity	Path exercised	Interpretation
Q1 – total replaced	0.674	Aggregation	Bimodal across runs
Q9 – frequency after 2024	0.670	Regulation retrieval	Underrated by cosine (see §5.4)
Q4 – panel history	0.619	12-column lookup	Long output; one empty run
Q8 – replaced >1 time	0.469	Multi-column boolean sum	Worst – negative ground truth

5.2.1 The strong end: definitional retrieval

The highest-scoring question, Q7 ("What do the defect grades mean in the Polito regulation?"), reached a mean fidelity of 0.891 and never dropped below 0.859 in any run. Its strength has a clear explanation. The answer is retrieved almost verbatim from Section 9 of the regulation, which is a compact, self-contained definition of the 0-to-4 grading scale. The retrieval step returns the correct passage, and the model is asked only to relay it, so the answer aligns closely with the expert key that was written from the same section. Q2 (the panel count) behaves similarly for a different reason: it is a single, unambiguous aggregation over the register that returns one number, leaving little room for the answer to diverge from the key. The pattern matches Phase 1: the system performs best where the target answer is closed and canonical.

5.2.2 The empty cell at question 4, run 7

The darkest cell in Figure 5.2, the 0.000 at question 4, run 7, is qualitatively different from every other low value in the matrix, and reading it correctly is essential. Question 4 asks for the full 2017–2025 inspection history of a single panel across twelve period columns, which is the longest single output the system is asked to produce. In run 7 the model returned an empty answer: on the CPU-only runtime used throughout this study, the generation of this long tabular response exceeded the per-question time limit, and the empty result was recorded and deliberately scored 0.0, because an empty string has no meaningful embedding and a missing answer is a genuine system outcome rather than a value to be excluded.

This is therefore a cell-wise, operational failure, not a reasoning failure and not a property of the question. The other nine runs of question 4 returned complete histories scoring between 0.651 and 0.712, and the empty run is the only reason question 4's mean is dragged to 0.619. It is also the sole reason run 7 has the lowest per-run mean of the whole experiment; if the empty cell is set aside, run 7's mean rises to 0.702, in line with the other runs. The correct conclusion is not that the model cannot answer question 4 but that a long, slow generation on constrained hardware can occasionally time out. It is an argument for a more

generous time budget, streamed or paginated output, or faster inference, and, at the interface level, for distinguishing an empty response from a valid answer that reports no records.

5.2.3 The pale row at question 8

Question 8, “Which panels were replaced (grade 4) more than once between 2017 and 2025?”, is the weakest question in the fidelity matrix, with a mean of 0.469 and individual cells falling as low as 0.296 (run 7) and 0.316 (run 9). Unlike question 4, this is a genuine row-wise pattern: every run scores poorly, so the cause lies with the question, not with any single execution. There are two contributing factors, and it is important to separate them because only one of them reflects a real limitation of the system.

The first and dominant factor is the nature of the correct answer. The ground truth for question 8 is a negative: no panel was ever graded 4 in more than one inspection period, because all 150 grade-4 events belong to distinct panels. The correct answer is therefore a short statement of absence — in essence, “none” with a brief justification. Cosine similarity is poorly suited to scoring such answers. A short, correct statement of absence shares little surface content with any reference wording, and two equally correct phrasings of “no panel meets this condition” can embed far apart. The low fidelity score is thus, to a large extent, a measurement artefact of using an embedding metric on an empty-set answer, and Section 5.4 shows that the manual review reclassifies most of these same answers as correct. The second, smaller factor is a real difficulty: question 8 requires the hardest computational expression in the set, a boolean sum across the twelve period columns to count how often each panel reached grade 4, and this multi-column operation is more error-prone than a single-column count. Question 8 therefore combines a metric that penalises the correct answer with the query most likely to be written incorrectly, which together explains the pale row.

5.3 Self-consistency across runs

Fidelity measures agreement with an external reference. It cannot, on its own, distinguish a system that is reliably almost-right from one that is erratic, and it penalises correct answers that are simply longer or differently phrased than the short expert key. A second measure was therefore computed: self-consistency, the cosine similarity between every pair of runs for the same question. Ten runs yield 45 unique pairs per question and 450 comparisons in total. Because both texts in each comparison are full model answers of comparable length, this measure asks a different question — not “is the answer correct?” but “does the system give the same answer when asked again?”. Figure 5.3 shows the pairwise grid for each question and Table 5.3 reports the mean agreement.

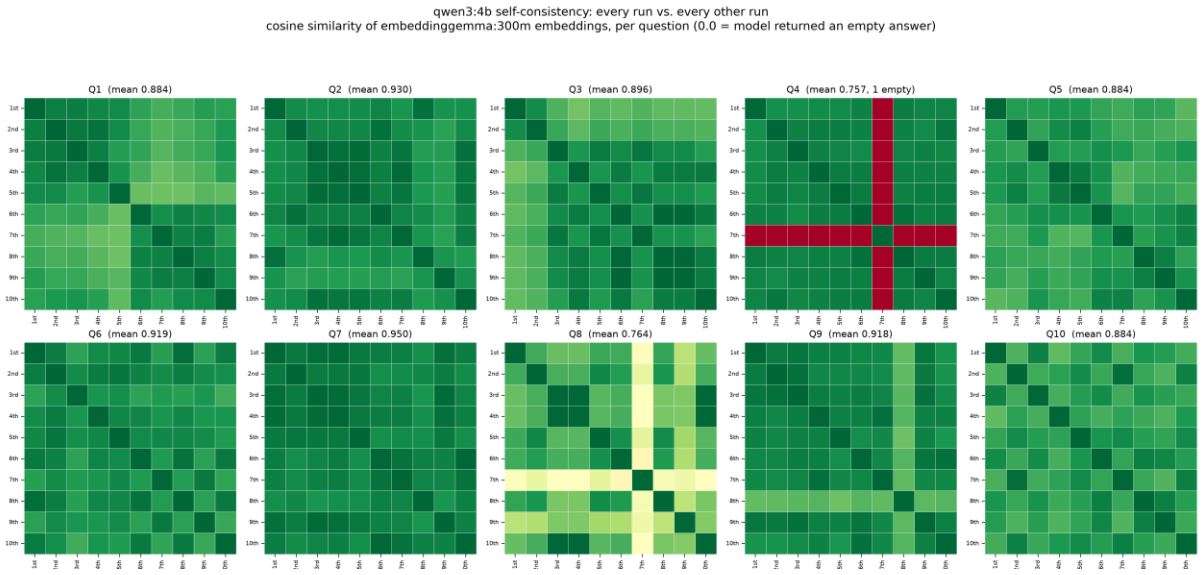


Figure 5.3. *Self-consistency across repeated runs, based on pairwise cosine similarity for each question.*

Table 5.3. *Self-consistency across 10 runs (45 run pairs per question), ordered from most to least consistent*

Question	Mean agreement	Min pair	Spread
Q7	0.951	0.903	0.096
Q2	0.931	0.877	0.115
Q6	0.919	0.850	0.136
Q9	0.918	0.788	0.196
Q3	0.896	0.778	0.222
Q1	0.884	0.787	0.188
Q5	0.884	0.816	0.145
Q10	0.884	0.808	0.177
Q8	0.764	0.480	0.521
Q4	0.757	0.000	0.984

Averaged over the ten questions, the mean pairwise agreement was 0.879, appreciably higher than the mean fidelity of 0.708. The two measures are related but not interchangeable, with a correlation of r = 0.82 across the ten questions. The gap between them is the single most important result of the evaluation: the prototype is more repeatable than it is accurate. When the same question is asked again, the system tends to return a very similar answer, but that answer is not always fully correct. This is, on balance, an encouraging property for a maintenance tool. A stable systematic error can be investigated and corrected through changes to the prompt, the tool descriptions, or the data handling, because it reproduces on

demand. An erratic error that appears and disappears between identical requests is far harder to diagnose and would be far more troubling in a system meant to inform maintenance decisions.

The grid also reinforces the row-versus-cell reading. Q7 is the most consistent question (0.951), almost uniformly dark green, mirroring its position as the most faithful; its answers are both correct and stable because they relay a fixed passage. The two least consistent questions are the same two that were weakest on fidelity, but for instructively different reasons that the grid makes visible. Question 4's low consistency (0.757) is produced almost entirely by a red cross — the seventh row and seventh column — where the empty run 7 disagrees maximally with every other run. Remove that one operational failure and question 4 is highly consistent, which confirms that its weakness is a single bad cell rather than a genuine instability. Question 8's low consistency (0.764) is a broader pale band with a minimum pair of 0.480, showing that the model expresses its short negative answer in genuinely different ways from run to run — again a consequence of the answer being a brief statement of absence that embedding similarity handles poorly, rather than of the underlying finding changing.

5.4 Manual four-class evaluation

Both automatic measures rely on embedding similarity, which is only an indirect proxy for factual correctness. To measure correctness directly, each of the 100 answers was reviewed by hand and assigned to one of four classes under a fixed priority rule: Error (a blank answer, non-answer, or surfaced tool failure), Hallucination (an answer containing unsupported invented detail), Wrong (an incorrect or materially incomplete answer), or Correct (an answer that materially matches the key). Each answer was classified by its most serious problem, so an otherwise correct answer that introduces an unsupported detail is counted as a hallucination. Figure 5.4 presents the resulting matrix and its summary, and Table 5.4 gives the totals.

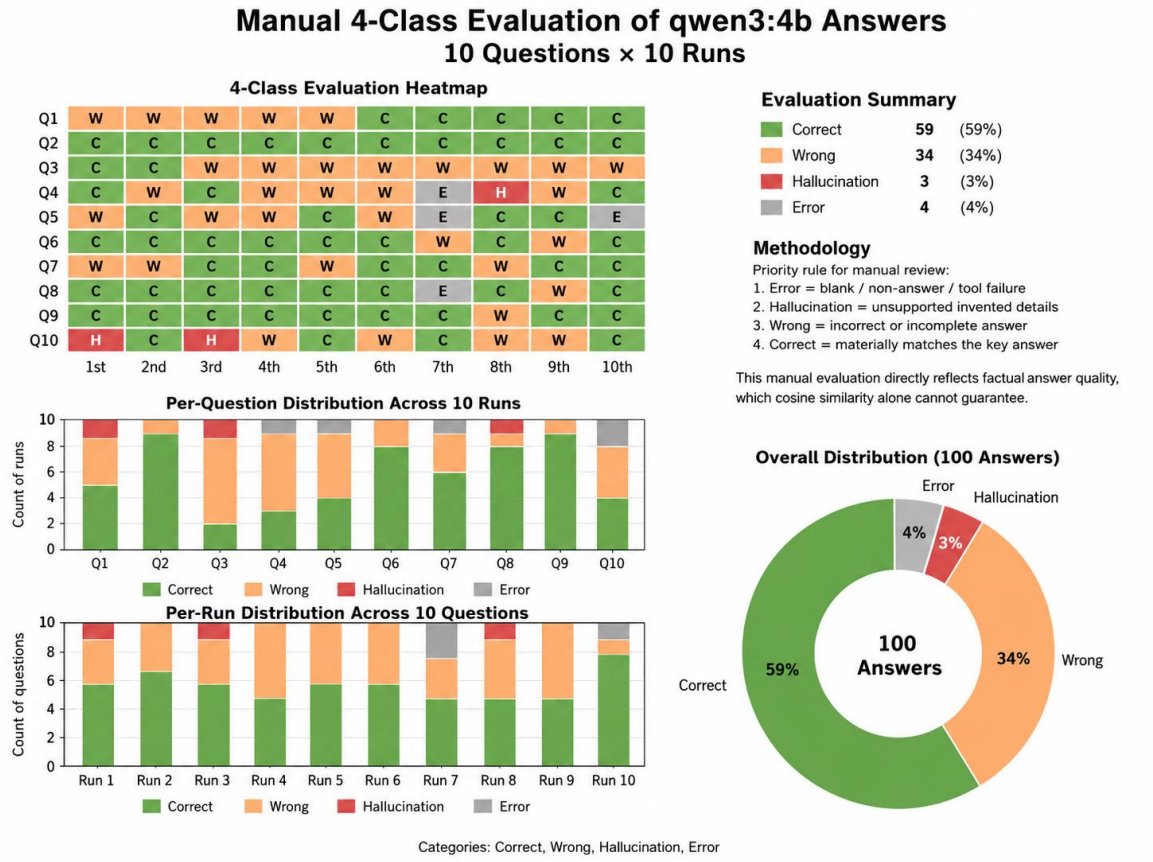


Figure 5.4. *Manual four-class evaluation of 100 answers, showing Correct, Wrong, Hallucination, and Error outcomes by question and run.*

Table 5.4. *Outcome of the manual four-class evaluation (100 answers)*

Class	Count	Share	Meaning
Correct	59	59%	Materially matches the key answer
Wrong	34	34%	Incorrect or materially incomplete
Hallucination	3	3%	Contains unsupported invented detail
Error	4	4%	Blank output or surfaced tool failure

Of the 100 answers, 59 were factually correct and 34 were wrong or materially incomplete. The two higher-risk categories were rare but not absent: three answers hallucinated detail and four were errors. Their placement in the matrix is meaningful. The four errors correspond exactly to the empty or failed runs already visible in the fidelity heat map — the timed-out question 4 in run 7, two failed runs of question 5, and one failed run of question 8 — confirming that these are operational rather than reasoning faults. The three hallucinations are concentrated where the model added context that had not been retrieved, at

question 10 (runs 1 and 3) and one run of question 4. For a safety-relevant maintenance application these two categories matter more than their small counts suggest, because a confidently invented detail or a silent failure is more dangerous than a visibly incomplete answer, and this is a direct argument for the provenance display and human-confirmation steps built into the framework.

5.4.1 Where correctness and cosine similarity disagree

The manual review is most valuable where it disagrees with the automatic scores, because those disagreements reveal the limits of embedding-based evaluation and, in several cases, rehabilitate the system's actual performance. Three questions illustrate the two opposite directions of error.

The clearest case is question 8. It was the worst question on fidelity, at 0.469, yet under manual review it is one of the better-answered questions: in eight of the ten runs the model correctly reported that no panel had been graded 4 in more than one period, with one wrong run and one operational error. The cosine metric had heavily penalised a short, correct statement of absence, exactly as Section 5.2.3 anticipated. Once read by a human, the same answers are seen to be right. Question 9 shows the same underrating more mildly: with a middling fidelity of 0.670 it is nonetheless correct in nine of ten runs, because its regulation-based answer is accurate but longer and more discursive than the compact expert key it was scored against.

Question 3 shows the opposite and more concerning direction. It scored a healthy fidelity of 0.783, the third highest in the set, but manual review found only two of its ten answers to be correct; the rest were wrong. Here the model produced fluent, regulation-styled prose about the Turin building framework that resembled the wording of the key closely enough to earn a high embedding score while being factually off. This is the failure mode that fluency-based metrics are least able to catch and that matters most in a maintenance setting: a confident, well-phrased, wrong answer. Question 1 completes the picture from a third angle. Its fidelity is bimodal — roughly 0.61 in the first five runs and 0.74 in the last five — and the manual grid explains why: five of its runs were judged wrong and five correct, a genuine stochastic split in correctness rather than a mere difference in wording.

Taken together, these disagreements reorder the difficulty ranking that cosine similarity alone would suggest. Embedding fidelity systematically underrates short and negative answers (questions 8 and 9) and can overrate fluent but incorrect ones (question 3). The practical lesson is that no single automatic score should be trusted as a verdict on a maintenance answer, and that the three measures used here are complementary: fidelity locates questions worth examining, self-consistency shows whether a problem is stable or erratic, and manual review delivers the factual verdict that the other two only approximate.

5.5 Tool use and the composition limitation

Alongside every answer, the evaluation harness recorded an independent execution trace: the tool calls that actually occurred, including the retrieved passage labels and the exact pandas expression executed against the register. Because this trace comes from the running system rather than from the wording of the answer, it provides an objective record of how each result was obtained and makes it possible to separate retrieval behaviour from generation behaviour.

Within each single question the routing was appropriate. The four regulation questions (Q3, Q7, Q9 and Q10) were answered through semantic retrieval over the document collection, and the six register questions (Q1, Q2, Q4, Q5, Q6 and Q8) were answered through a direct pandas query over the 5,496-row table. The agent generally selected the correct single tool for the task, which is why the retrieval-grounded definitional questions score as well as they do. The decisive finding, however, is a limitation: across all 100 recorded answers, the agent never called both tools within a single question. It did not, in any run, retrieve a value from the register and then fetch the regulation passage that explains it, or the reverse.

For the ten evaluation questions this did not cause failures, because each was designed to be answerable with a single tool. It nonetheless exposes a genuine ceiling on the prototype's capability. Many real maintenance questions are inherently mixed — for example, "how many panels are at grade 3, and what does the regulation require for that grade?" requires a computation over the register and a retrieval from the regulation in the same answer. The 4-billion-parameter model did not compose its two tools to serve such questions, and this is the principal reason the interface mock-up in Chapter 4 that shows a combined answer is described there as a design target rather than measured behaviour. The limitation can be addressed without abandoning local execution, either by a deterministic two-stage controller that calls both tools and then asks the model only to synthesise their outputs, or by repeating the evaluation with a larger local model; both are identified in Section 4.7 as routes for future work.

5.6 Discussion

In conclusion, the three measures give a consistent and defensible account of the prototype. Its answers are stable across repeated runs (mean self-consistency 0.879), moderately faithful to the expert key when judged by embedding similarity (mean fidelity 0.708) and correct in a clear majority of cases when judged by a human (59 of 100). The distance between these numbers is not a contradiction but the central result: the system is more repeatable than accurate, and it is somewhat more accurate than a fluency-based metric alone would suggest, because that metric penalises the short and negative answers the system often gets right.

The evaluation matrices also localise where effort should be directed. Weakness in this system is overwhelmingly organised by question, not by run. There is no weak column in either figure 4.3, which means the pipeline does not drift or degrade globally; instead, specific questions carry specific, explicable difficulties. Some of those difficulties are artefacts of measurement that disappear under human review, as with the negative ground truth of question 8. Some are operational and hardware-bound, as with the single timed-out generation at question 4, run 7, which argues for engineering remedies rather than model changes. And some are real reasoning or retrieval faults, as with the fluent-but-wrong answers to question 3, which argue for better grounding and verification. Because each kind of weakness leaves a different signature in the matrices — a weak row, a weak cell, or a divergence between automatic and manual scores — the evaluation not only measures performance but also tells a developer what to fix.

These results answer the thesis's central research question in a qualified but clear way. An LLM-based interface can connect heterogeneous maintenance information to BIM and return grounded, mostly correct answers from natural-language questions while running entirely on local infrastructure, which satisfies the confidentiality requirement for safety-relevant condition data. At the same time, a one-third rate of wrong or incomplete answers, the presence of occasional hallucinations and errors, and the absence of reliable tool composition all place a firm boundary on how the system may be used. At its present level of performance, it is a decision-support assistant, not an autonomous authority. This directly justifies the governance design of Chapter 4, in which the model proposes a structured record but a human inspector or facility manager reviews and approves it before any value is written back to the BIM model, and in which the provenance of every answer — the sources retrieved and the query executed — is exposed so that the user can verify how a result was reached. The evaluation shows both that such a system is feasible under realistic institutional constraints and precisely why humans remain in the loop.

6. Conclusions and Future Work

This chapter brings together the main findings and their implications for LLM-based interfaces in BIM-enabled maintenance. Because the prototype was developed under practical constraints—local execution, limited hardware, mixed data sources, and human approval of BIM updates—the conclusions focus on what was demonstrated under those conditions and on the work still required for broader use.

6.1 Conclusions

The research question asked how an LLM-based interface can connect heterogeneous building-maintenance information to BIM while maintaining computational reliability and human control over model updates. The results show that this connection is feasible, but that it depends on treating different kinds of information differently. Regulatory text and technical documents can be retrieved semantically, whereas the inspection register requires direct computation when the answer depends on filtering, counting, grouping, or a complete history panel. Keeping these two paths separate was necessary to avoid using vector retrieval for operations that require exact table-wide results.

The model-selection phase showed only small differences between the four local candidate models. qwen3:4b achieved the highest mean similarity to the expert reference answers, at 0.8284, while also offering a smaller 4B resource footprint and native tool calling. It was therefore a practical choice for the prototype rather than a statistically superior model. This distinction matters because the later evaluation shows that general BIM and facility-management knowledge is only one part of the problem. The more important question is whether the selected model can use project-specific evidence and the available tools correctly.

Inside the complete retrieval-augmented system, the prototype produced a mean answer fidelity of 0.708 and a mean self-consistency of 0.879 across 100 responses. The manual review classified 59 of the 100 answers as correct. Taken together, these measures show a system that is more repeatable than accurate. The repeated-run matrices were generally stable, and most weaknesses were associated with particular questions rather than with entire runs. At the same time, the manual review showed why a fluent or semantically similar answer cannot be assumed to be correct. Short negative answers were sometimes underrated by cosine similarity, while some well-phrased but factually wrong answers received relatively strong similarity scores.

The execution trace added a further layer to this evaluation by recording the tool calls and the Pandas' expressions independently of the final wording of the answer. For the ten evaluation questions, the agent

generally selected the appropriate single tool. However, it never composed the two tools within the same question. This places a clear boundary on the present system: questions that require both an exact result from the register and an interpretation from the regulation are not yet reliable without a more controlled orchestration strategy.

The second direction of the framework, from inspection information back to BIM, was demonstrated through the Dynamo workflow. The unique panel code provided the common key between the Revit curtain-panel instances and the inspection register, allowing validated condition information to be transferred to the corresponding BIM elements and exported again to the register. This closes the information loop at prototype level and addresses a common weakness of maintenance workflows, in which inspection records continue to evolve outside the BIM model. The important condition is that the LLM does not act as an autonomous editor: the proposed record must be checked and approved before any change is committed.

Overall, the study supports the use of a local LLM-BIM interface as a decision-support layer for maintenance rather than as an independent authority. The prototype can reduce the effort required to navigate several sources and can connect natural-language questions to building records, regulatory documents, and BIM elements. Its present error rate and tool-composition limitation, however, make provenance, verification, and human approval essential parts of the workflow rather than optional safeguards.

6.2 Main Contributions

The first contribution is a source-aware hybrid architecture for maintenance information. Instead of placing every project source in the same vector database, the framework preserves the structure of the inspection register and queries it computationally, while using retrieval-augmented generation for prose documents. This design follows the nature of the data rather than forcing all information through one retrieval mechanism.

The second contribution is the evaluation approach. Answer fidelity, self-consistency, manual factual review, and execution tracing were considered separately. This made it possible to distinguish a stable but wrong answer from a stochastic failure, a metric artefact, and an operational timeout. In a maintenance context, that distinction is useful because each failure type points to a different corrective action.

The third contribution is the explicit verification of tool use. The evaluation did not rely only on what the model claimed to have done; the harness recorded the retrieved sources and the exact data query that

was executed. This exposed the absence of multi-tool composition even when the final answer itself appeared plausible, and it provides a more transparent basis for assessing an agent that may inform maintenance decisions.

The fourth contribution is a controlled two-way link between maintenance records and BIM. The workflow uses the verified panel code as a shared asset identifier, transfers inspection information through Dynamo, and keeps the update subject to human confirmation. The use of BIM instance parameters together with IFC-oriented interoperability principles also keeps the proposed process compatible with a wider information-management workflow rather than tying it only to a conversational interface.

6.3 Limitations

The conclusions should be read within the limits of the study. The complete Phase 3 evaluation was carried out only with qwen3:4b, even though the Phase 1 benchmark showed relatively small differences among the candidate models. The case study also concerns one building, one facade system, and one inspection register. The results therefore demonstrate the behaviour of the framework in this case rather than its performance across different portfolios, asset classes, or organisational settings.

The evaluation metrics also have limitations. Cosine similarity is useful for comparing large sets of free-text answers, but the results in Chapter 5 show that it can undervalue short correct answers and overvalue fluent incorrect ones. The manual classification reduced this problem but was performed by a single reviewer, so no inter-rater agreement was measured. A stronger validation would combine automatic metrics with at least one additional independent factual review.

Two implementation limits are especially important. First, the agent did not reliably combine the document-retrieval and register-query tools in one request. Second, the write-back path was implemented and demonstrated in Dynamo, but the complete sequence in which a free-text field observation is converted by the LLM into a validated structured record was not evaluated experimentally. The restricted pandas execution environment is also appropriate only for the local research prototype; a multi-user deployment would require a stronger execution boundary.

6.4 Future Work

The immediate priority is to improve orchestration for mixed maintenance questions. A deterministic two-stage controller could execute the register query and the document retrieval explicitly before asking the model to synthesise the two outputs. This would remove the need for a small model to decide whether a second tool call is required. The same evaluation should then be repeated with other local models,

including a larger model where hardware permits, to determine whether the limitation comes mainly from model capacity or from the agent design.

A second direction is broader validation. The question set should be expanded beyond the ten case-study items and should include more mixed queries, ambiguous wording, missing-data cases, and tasks involving different component types. Applying the framework to additional buildings would test whether the panel-code strategy and the hybrid data architecture remain effective when asset schemas, inspection formats, and maintenance rules are less uniform.

The write path also requires a dedicated end-to-end evaluation. A useful next experiment would start from a controlled set of field observations with known target values, convert each observation into the proposed inspection schema, and score every field by exact match before any BIM update is allowed. Repeated trials would make it possible to measure both accuracy and stability, while a second reviewer could assess cases where the observation itself is ambiguous. This would provide evidence for the part of the workflow that currently depends most strongly on human confirmation.

Finally, a production-oriented version would require stronger security, clearer roles, and interface testing with facility-management staff. A web interface is the most suitable primary channel for tables and provenance, while mobile capture could be added where institutional data-governance requirements permit. The central principle should remain unchanged: the language model may assist access and coordination, but the authoritative building record must remain traceable, verifiable, and under human control.

References

[1] Kim, N. (2025). *Automated robotic planning for corrective building maintenance using LLMs and BIM-driven 3D scene graphs* [Master's thesis, Technical University of Munich].

[2] Li, Y., Ji, M., Chen, J., Wei, X., Gu, X., & Tang, J. (2025). A large language model-based building operation and maintenance information query. *Energy and Buildings, 334*, 115515. https://doi.org/10.1016/j.enbuild.2025.115515

[3] Park, I., Kim, Y., Kim, K., & Chin, S. (2026). BIM–LLM integration in AECO workflows: Applications, validation, and future directions. *Automation in Construction, 187*, 106905. https://doi.org/10.1016/j.autcon.2026.106905

[4] Xue, H., Guo, P., Ma, J., Lin, J.-R., & Cheng, J. C. P. (2025). Exploring automated query understanding for BIM information retrieval: A multiple-agents collaboration framework leveraging a large language model. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.5142300

[5] Liu, B., & Chen, H. (2025). BIMCoder: A comprehensive large language model fusion framework for natural language-based BIM information retrieval. *Applied Sciences, 15*(14), 7647. https://doi.org/10.3390/app15147647

[6] Nithyanantham, B. K., Sesterhenn, T., Nedungadi, A., Garijo, S. P., Zenkner, J., Bartelt, C., & Lüdtke, S. (2025). *MCP4IFC: IFC-based building design using large language models* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2511.05533

[7] Heaton, J., Parlikad, A. K., & Schooling, J. (2019). Design and development of BIM models to support operations and maintenance. *Computers in Industry, 111*, 172–186. https://doi.org/10.1016/j.compind.2019.08.001

[8] Inojosa, L. da S. P., & Vilanova, K. de A. (2023). BIM interoperability in the maintenance planning process for existing buildings. *Journal of Building Pathology and Rehabilitation, 8*, 45. https://doi.org/10.1007/s41024-023-00292-4

[9] Ehab, A., Mahdi, M. A., & El-Helloty, A. (2024). BIM maintenance system with IoT integration: Enhancing building performance and facility management. *Civil Engineering Journal, 10*(6), 1953–1973. https://doi.org/10.28991/CEJ-2024-010-06-015

[10] Zheng, Z., Marié, S., & Kubler, S. (2026). Retrieval-augmented LLM with structured sampling for building management systems point tagging under minimal context. *Automation in Construction*. In press.

[11] Madireddy, S., Gao, L., Din, Z. U., Kim, K., Senouci, A., Han, Z., & Zhang, Y. (2025). Large language model-driven code compliance checking in building information modeling. *Electronics, 14*(11), 2146. https://doi.org/10.3390/electronics14112146

[12] Lee, G., Jang, S., & Hyun, S. (2024). *A generalized LLM-augmented BIM framework: Application to a speech-to-BIM system* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2409.18345

[13] Akhavian, R., Amani, M., Mootz, J., Ashe, R., & Beheshti, B. (2025). *Building information models to robot-ready site digital twins (BIM2RDT): An agentic AI safety-first framework* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2509.20705

[14] Lin, Q., & Ren, G. (2025). Automating interoperable reasoning for BIM-based regulatory and performance knowledge using the Semantic Web: A systematic review [Online manuscript]. *Semantic Web Journal*. https://www.semantic-web-journal.net/system/files/swj3925.pdf

[15] Ratul, A. K., Acharjee, S., Park, S., & Sakib, M. N. (2025). *Sketch2BIM: A multi-agent human–AI collaborative pipeline to convert hand-drawn floor plans to 3D BIM* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2510.20838

[16] Bagasi, O., Nawari, N. O., & Alsaffar, A. (2025). BIM and AI in early design stage: Advancing architect–client communication. *Buildings, 15*(12), 1977. https://doi.org/10.3390/buildings15121977

[17] Xu, D. (2020). Summary of operation and maintenance with BIM technology for large commercial complex architecture. In *Proceedings of the 3rd International Conference on Education, Economics and Management Research (ICEEMR)* (pp. 462–465). https://doi.org/10.2991/assehr.k.191221.111

[18] Yang, L., & Su, R. (2025). From machine learning-based to LLM-enhanced: An application-focused analysis of how Social IoT benefits from LLMs. *IoT, 6*(2), 26. https://doi.org/10.3390/iot6020026

[19] Chin, D., Wang, Y., & Xia, G. (2024). *Human-centered LLM-agent user interface: A position paper* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2405.13050

[20] Khan, A. A., Hasan, M. T., Kemell, K. K., Rasku, J., & Abrahamsson, P. (2024). *Developing retrieval-augmented generation (RAG)-based LLM systems from PDFs: An experience report* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2410.15944

[21] Gan, A., Yu, H., Zhang, K., Liu, Q., Yan, W., Huang, Z., Tong, S., & Hu, G. (2025). *Retrieval-augmented generation evaluation in the era of large language models: A comprehensive survey* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2504.14891

[22] Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., & Abdelrazek, M. (2024). Seven failure points when engineering a retrieval-augmented generation system. In *Proceedings of the IEEE/ACM International Conference on AI Engineering—Software Engineering for AI (CAIN)* (pp. 1–6). https://doi.org/10.1145/3644815.3644945

[23] Lála, J., O'Donoghue, O., Shtedritski, A., Cox, S., Rodriques, S. G., & White, A. D. (2023). *PaperQA: Retrieval-augmented generative agent for scientific research* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2312.07559

[24] Godinez, A. (2025). *HySemRAG: A hybrid semantic retrieval-augmented generation framework for automated literature synthesis and methodological gap analysis* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2508.05666

[25] Klesel, M., & Wittmann, H. F. (2025). Retrieval-augmented generation (RAG). *Business & Information Systems Engineering, 67*(4), 551–561. https://doi.org/10.1007/s12599-025-00945-3

[26] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-T. (2020). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 6769–6781). https://doi.org/10.18653/v1/2020.emnlp-main.550

[27] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (pp. 3982–3992). https://doi.org/10.18653/v1/D19-1410

[28] Steck, H., Ekanadham, C., & Kallus, N. (2024). *Is cosine similarity of embeddings really about similarity?* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2403.05440

[29] Cer, D., Yang, Y., Kong, S.-Y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil, R. (2018). *Universal sentence encoder* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1803.11175

[30] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)* (pp. 4171–4186). https://doi.org/10.18653/v1/N19-1423

[31] Vaswani, A., et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 5998–6008).

[32] Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. In *Proceedings of the BlackboxNLP Workshop* (pp. 276–286). https://doi.org/10.18653/v1/W19-4828

[33] van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*, 2579–2605.

[34] Lee, K.-H., Fischer, I., Wu, Y.-H., Marwood, D., Baluja, S., Schuurmans, D., & Chen, X. (2025). *Evolving deeper LLM thinking* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2501.09891

[35] Garrido-Merchán, E. C. (2025). *Information-theoretic Bayesian optimization: Survey and tutorial* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2502.06789

Appendix A: Model selection benchmark questions

The twenty questions below make up the Phase 1 benchmark used to select the generation model (Section 4.2). Each was submitted to all four candidate models under identical conditions, and each model answer was compared with an expert reference answer by cosine similarity, as described in Section 4.2.3. The questions span three kinds of knowledge: definitional items that probe core concepts, applied items about using BIM in day-to-day facility management, and integration items covering digital twins and IoT.

Table A.1. *The BIM and facility-management questions used in the model-selection benchmark*

No.	Question	Category
1	What is Building Information Modelling (BIM)?	Definitional
2	How does BIM support facility management activities?	Applied
3	What information is typically included in a BIM model for FM?	Definitional
4	What are the main benefits of using BIM during the operations phase of a building?	Applied
5	How does BIM improve maintenance planning?	Applied
6	What are common challenges when implementing BIM for facility management?	Applied
7	What is the difference between BIM used in construction and BIM used in facility management?	Definitional
8	How can BIM help reduce operational costs in a building?	Applied
9	What role does a facility manager play in defining BIM requirements?	Applied
10	What is COBie and how is it used in FM?	Definitional
11	How can BIM improve asset tracking in a building?	Applied
12	What software tools are commonly used for BIM-based facility management?	Applied
13	How does BIM support energy management in facilities?	Applied
14	What types of data need to be updated in a BIM model during building operations?	Applied
15	How can BIM improve space management and space utilization analysis?	Applied
16	What is a digital twin and how does it relate to BIM and FM?	Integration
17	Why is data accuracy important when using BIM for FM?	Applied
18	How can BIM assist with emergency planning and response?	Applied
19	What training do facility managers need to effectively use BIM?	Applied
20	How does integrating IoT with BIM enhance facility operations?	Integration