The European Master in Building Information Modelling is a joint initiative of:

Universidade do Minho

UNIVERZA V LJUBLJANI

AI-Driven Assistant for LOIN in BIM Workflows

Pablo Ignacio S. Camus Rojas

BIM A+

UMinho | 2025

Universidade do Minho
Escola de Engenharia

Pablo Ignacio S. Camus Rojas

AI-Driven Assistant for LOIN in BIM Workflows

BIM A+ European Master in Building Information Modelling

SEPTEMBER 2025

Universidade do Minho
Escola de Engenharia

Pablo Ignacio S. Camus Rojas

AI-Driven Assistant for LOIN in BIM Workflows

Master Dissertation
European Master in Building Information Modelling

Work conducted under supervision of:
José Luis Duarte Granja
Mohamad El Sibaii

September, 2025

AUTHORSHIP RIGHTS AND CONDITIONS OF USE OF THE WORK BY THIRD PARTIES

ACKNOWLEDGEMENTS

The development of this work could not have been undertaken without the scholarship granted by BIM A+. My profound gratitude is extended to the program and its members for this opportunity.

At the same time, I would like to express my deepest gratitude to my thesis supervisor, Mr. José Luís Duarte Granja, and my co-supervisor, Mr. Mohamad El Sibaii, for their unwavering commitment to the development of this research. Their guidance and insights were instrumental in shaping and enriching the content of this work.

To my parents, Michael Camus Davila and Paula Rojas Martin, who inspired me to pursuit my academic interests. To my grandparents, Nemesio Camus Pizarro (†), Magaly Davila Cornejo, Manuel Rojas Troncoso and Emilia Martin Garcia, to whom I owe a lasting curiosity for the different fields of the academic world. To my siblings, José, Francisco and M. Trinidad, for their invaluable companionship. To my cousins, uncles and all relatives, for their encouragement and support throughout this journey.

A special acknowledgement is due to my partner, Karyna Saifudinova, the editor of this work, whose resilience and steadfast support have been immeasurable throughout this adventure.

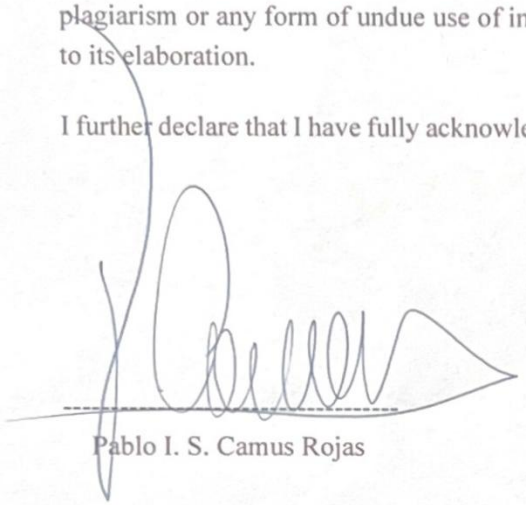
I am deeply grateful to Undurraga Deves Arquitectos, and in particular to Mr. Cristián Undurraga Saavedra, for allowing me to continue with the exercise of my profession through the development of the thesis.

Lastly, heartfelt thanks go to my friends. Their companionship, encouragement and patience throughout this process have been invaluable.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.



Pablo I. S. Camus Rojas

RESUMO

A especificação rigorosa do Nível de Necessidade de Informação (LOIN) é crítica para a qualidade da informação e a interoperabilidade nos processos BIM, embora a autoria do LOIN permaneça predominantemente manual, suscetível a ambiguidade terminológica e a desalinhamentos com as normas internacionais. Esta dissertação investiga uma abordagem baseada em IA para automatizar a documentação do LOIN, combinando Geração Aumentada por Recuperação (RAG) com uma estruturação alinhada a normas, no enquadramento normativo das ISO 19650, ISO 7817-1, ISO 29481-1 e ISO 23387. Uma metodologia exploratória e iterativa sustenta o trabalho, culminando num protótipo conversacional e num roteiro de desenvolvimento modular.

Esta dissertação tem como objetivo avaliar a eficácia dessa abordagem através de experiências comparativas. O estudo realiza um benchmark de modelos de linguagem generalistas sem orientação adicional, avaliando a precisão terminológica e a conformidade estrutural com o LOIN, e avalia um protótipo que integra recuperação de passagens ancorada nas normas, estratégias de segmentação (chunking) e indexação, e esquemas de estruturação de requisitos para orientar a geração. Resultados preliminares indicam melhoria da conformidade com o LOIN, maior clareza organizacional e maior estabilidade entre execuções, quando comparado com LLMs utilizados de forma isolada.

Trabalho futuro inclui validação baseada em regras, aprofundamento da ligação a dicionários de dados do domínio e estudos com utilizadores em contextos reais de projeto.

Palavras chave: BIM, LOIN, LLM, Retrieval-Augmented Generation, Standards.

ABSTRACT

Accurate specification of the Level of Information Need (LOIN) is critical to information quality and interoperability in BIM processes, yet LOIN authoring remains predominantly manual, prone to terminological ambiguity and inconsistent alignment with international standards. This dissertation investigates an AI-driven approach to automate LOIN documentation by combining Retrieval-Augmented Generation (RAG) with standards-aligned structuring within the normative framework of ISO 19650, ISO 7817-1, ISO 29481-1 and ISO 23387. An exploratory, iterative methodology underpins the work, culminating in a conversational prototype and a modular development roadmap.

This thesis aims to evaluate the effectiveness of that approach through comparative experiments. The study benchmarks mainstream large language models without additional guidance, assessing terminology accuracy and structural compliance with LOIN, and evaluates a prototype that integrates standards-grounded passage retrieval, tailored chunking and indexing strategies, and requirement-structuring schemes to steer generation. Preliminary findings indicate improved compliance with LOIN, clearer structural organization and higher run-to-run stability compared with standalone LLMs.

Future work includes rule-based validation, deeper linkage to domain data dictionaries, and user studies in real project settings.

Keywords: BIM, LOIN, LLM, Retrieval-Augmented Generation, Standards.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

BIM	Building Information Modelling
AECO	Architecture, Engineering, Construction, and Operations
LOIN	Level of Information Need
AI	Artificial Intelligence
LLM	Larga Language Model
RAG	Retrieval Augmented Generation
ISO	International Organization for Standardization
LOD	Level of Detail
IDM	Information Delivery Manual
CDE	Common Data Environment
bsDD	BuildingSmart Data Dictionary
RDF	Resource Description Framework
IFC	Industry Foundation Classes
HVAC	Heating, Ventilation, and Air Conditioning
SPARQL	Simple Protocol and RDF Query Language
API	Application Programming Interface
MCP	Model Context Protocol
CLI	Command-Line Interface
SHACL	Shapes Constraint Language

1. INTRODUCTION

1.1. Context and Motivation

The use of *Building Information Modelling* (BIM) workflows has become increasingly common in the *Architecture, Engineering, Construction, and Operations* (AECO) sector. The use of BIM has significantly improved project coordination, the management of information, and the efficiency of it throughout different project stages.

A key aspect of successful BIM's framework implementation depends on the correct definition of the *Level of Information Need* (LOIN). LOIN documentation clearly outlines the exact alphanumerical and geometrical information required for a particular purpose at a specific milestone and by a particular stakeholder. This crucial step determines how efficiently data is encapsulated within the BIM workflow. However, creating and managing LOIN documentation is usually a manual and complex process. Hand-crafting lists, cross-referencing model elements, and interpreting dense standards can introduce inconsistencies, inaccuracies, and inefficiencies that detract from BIM's overall benefits.

Additionally, professionals who are less familiar with specialized information-management standards often struggle to generate accurate LOIN documentation. This difficulty can lead to uneven document quality, platform incompatibilities, and delayed approvals ultimately affecting the effectiveness and interoperability of the entire BIM workflow. Addressing these challenges is vital to ensure that every project phase delivers the right data, at the right time, to the right people.

1.2. Research Subject and Scope

To address these challenges, this thesis investigates the use of *Artificial Intelligence* (AI) and in particular the use of mainstream *Large Language Models* (LLMs) to automate the authoring of LOIN documentation. By embedding *Retrieval-Augmented Generation* (RAG) pipelines and complementary AI strategies, the research seeks to streamline the generation process, reduce manual effort, and improve the consistency and reliability of the output.

Furthermore, the thesis will explore the integration of ontology-based methods into these AI-driven approaches. This integration could enhance both the clarity and structure of the generated documentation by introducing a layer of semantic consistency grounded in formal domain knowledge. Additionally, the research will examine different implementation strategies, including existing state of the art LLMs and custom-built solutions, to identify effective methods for automating LOIN documentation.

1.3. Purpose and Objectives

The main purpose of this research is to develop and evaluate an automated, AI-driven approach for generating precise, consistent, and user-friendly LOIN documentation. The research aims to simplify the LOIN definition process by leveraging AI techniques, such as RAG, making it accessible to both trained BIM specialists and general AECO practitioners.

This thesis is guided by the following objectives:

- Develop and test an AI-based application capable of generating LOIN documentation using mainstream LLMs.
- Enable AECO professionals who lack extensive BIM-specific training to accurately produce standardized LOIN documentation, thus bridging the knowledge gap.
- Theoretically explore strategies for integrating ontology-based methods within AI-driven workflows, identifying potential directions for future development.

Ultimately, this research aims to improve BIM workflows by providing tools that democratize the creation of high-quality LOIN documentation, ensuring accuracy and clarity regardless of user's prior knowledge of detailed information management standards.

1.4. Assessment of Existing Studies and Gaps

Several international standards now provide robust frameworks for defining information requirements within BIM. The ISO 29481-1 Information Delivery Manual outlines processes for capturing exchange requirements, while ISO 7817-1 formalizes LOIN to specify both alphanumeric and geometric data granularity. Likewise, the ISO 19650 series establishes principles for information management across both project and asset delivery phases.

Despite these advances, the actual creation of LOIN documentation remains largely manual. Existing software and methodologies guide users through requirement definition but offer limited support for end-to-end automation or intelligent assistance.

Early work in semantic technologies and ontologies has shown promise for structuring domain knowledge, and recent developments with AI, particularly with LLMs, suggest opportunities for automating semantic tasks. At the same time, many of these tools present a steep learning curve, making them difficult for non-specialists to adopt.

Not a single solution today integrates standard-compliant LOIN definitions, RAG, and ontology-driven reasoning within a unified user-friendly interface. Moreover, systematic comparisons of different AI deployment strategies, cloud services, custom LLM endpoints, and local models are not yet available.

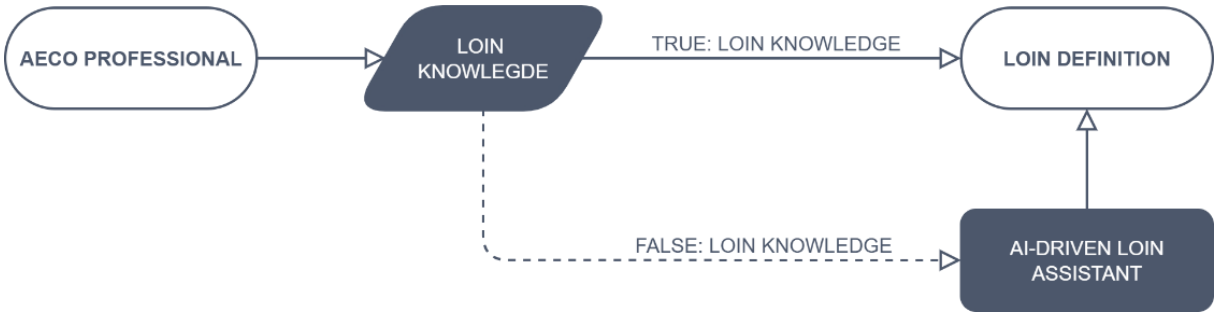


Figure 1 – Prototype's expected function

This thesis explores this gap by developing and evaluating an AI-driven application tailored to automate and enhance LOIN documentation in accordance with relevant ISO standards, while maintaining an intuitive user experience. Figure 1 illustrates the intended gap to cover by the developed prototype.

1.5. Research Methodology

This thesis adopts an exploratory methodology grounded in iterative prototyping. Each cycle is used to probe feasibility, surface constraints, and inform subsequent design choices, with targeted tests oriented to BIM information retrieval and ontology alignment.

The work began with a minimal RAG pipeline to retrieve ISO standard excerpts into cloud and local language models, assessing their capacity to surface BIM-relevant information. The pipeline was then extended in two directions: RAG combined with OpenAI models, and RAG combined with ontology alignment and data dictionaries. These trials revealed gaps in concept coverage and rigidity caused by strict term lists, which limited flexibility in assistant behavior.

As a consequence of these discoveries, the retrieval stack was redesigned to improve filtering and control the scope of question formation. This included the development of chunking strategies tailored to standards content and the gradual implementation of functionalities for semantic querying and LOIN drafting. The resulting steps and design decisions are detailed in Chapter 3.

The current prototype operates as a general AI assistant capable of deep semantic inquiry and guided LOIN drafting. A version-control appendix (Appendix 1) documents major iterations and design decisions prior to the main chapters, ensuring transparency of the development path.

1.6. Thesis Structure

This subsection outlines how the thesis is organized, highlighting how each chapter contributes to the exploratory development and assessment of AI-driven automation and ontology-based structuring for LOIN in BIM workflows.

- Chapter 1 Introduction: Presents the motivation, research questions, scope, and the exploratory methodology guiding the work.
- Chapter 2 State of the Art: Reviews LOIN and its three dimensions, the relevant standards landscape (ISO 7817-1, ISO 19650, ISO 29481-1, ISO 23387), and the role of ontologies and data dictionaries for semantic interoperability.
- Chapter 3 Methodology and Prototype Development: Details the exploratory roadmap: Phase 1 (RAG baseline; extensions with OpenAI; ontology alignment and data dictionaries), followed by Phase 2 (retrieval redesign, chunking strategies, and gradual implementation of functionalities for semantic querying and guided LOIN drafting).
- Chapter 4 Evaluation and Results: Describes datasets, metrics, and protocols; reports comparative results across iterations and models; and discusses validity and limitations relative to BIM information needs.
- Chapter 5 Conclusions and Future Work: Summarizes contributions, answers the research questions, and outlines the next steps for scaling and standardization.

This page is intentionally left blank

2. THEORETICAL FRAMEWORK AND STATE OF ART

This chapter outlines the theoretical framework and state of the art, covering LOIN and governing standards, AI/LLMs and RAG, and semantic strategies via ontologies and data dictionaries.

2.1. BIM Workflows and Information Management

The adoption of BIM in the AECO sector has transformed how building projects are conceived, managed, and delivered. BIM enables the creation of a shared digital representation of built assets that supports decision-making across the asset's lifecycle from design to demolition. It replaces fragmented document exchanges with a centralized, object-based model that integrates both graphical and alphanumeric data.

In this integrated environment, each model element encapsulates information about its physical characteristics, functional requirements, and relationships with other components. For instance, a wall object might contain not only geometric data but also fire-rating properties, acoustic performance values, and manufacturer specifications. This approach supports enhanced coordination, reduces errors, and facilitates automation of quality control processes (ISO19650-1, 2018).

However, the effectiveness of BIM largely depends on how information is specified and structured throughout the workflow. Projects must ensure that stakeholders receive the right information, at the right level of detail, and at the right time. This is where information requirements play a critical role. Without clearly defined expectations, models may either contain excessive, unused data, creating complexity and performance issues, or lack critical details required for specific tasks, such as cost estimation or compliance checking (ISO7817-1, 2024). Traditionally, information requirements were captured through narrative specifications or spreadsheets, often disconnected from the BIM environment. These methods are not only labor-intensive but also susceptible to inconsistencies, especially in projects with multiple contributors. Manual interpretation of requirements often leads to errors and omissions that compromise downstream uses of the model (ISO29481-1, 2016).

In this context, the concept of LOIN was introduced to formalize these expectations. LOIN specifies the minimum content (both geometric and alphanumeric) that must be included in the model at each project stage, tailored to the needs of specific actors. It serves as a foundation for requirement-driven modeling and verification, helping to avoid both over-modeling and under-modeling scenarios (ISO7817-1, 2024). Despite these formalizations, creating LOIN documentation remains a manual and error-prone task in most workflows. Few tools offer dynamic guidance or real-time validation against predefined requirements. This lack of automation poses a barrier for non-specialists and limits the broader adoption of structured information practices in BIM.

In summary, while BIM offers a robust framework for information-centric collaboration, its success relies heavily on well-defined, consistently applied information requirements. Automating the authoring of such requirements through AI, particularly using retrieval-based methods and formalized data structures, presents an opportunity to increase efficiency, accuracy, and accessibility across project teams.

2.2. Level of Information Need (LOIN): Concepts & Practices

The LOIN framework provides a structured method to define what information is required, when it is needed, and for whom it is relevant throughout a project's lifecycle. Formalized in the international standard ISO 7817-1, LOIN was developed to prevent over-specification and to ensure that only the necessary information is included in the BIM model at each project stage (ISO7817-1, 2024).

LOIN encompasses three dimensions of information: geometric information, alphanumeric information, and documentation. Geometric information includes the graphical representation and dimensions of an object, while alphanumeric information refers to structured properties such as material specifications, fire resistance, or energy ratings. Documentation includes references to external resources like standards, datasheets, or certificates (ISO7817-1, 2024).

Unlike earlier approaches that used ambiguous terms such as *Level of Detail* (LOD) or *Level of Development*, LOIN separates the purpose of the information from its format. Each requirement is defined in relation to a specific use case, such as design validation, cost estimation, or regulatory approval, and is tailored to the actor's needs and the project phase. This approach reduces ambiguity and increases model reliability for downstream users (ISO7817-1, 2024).

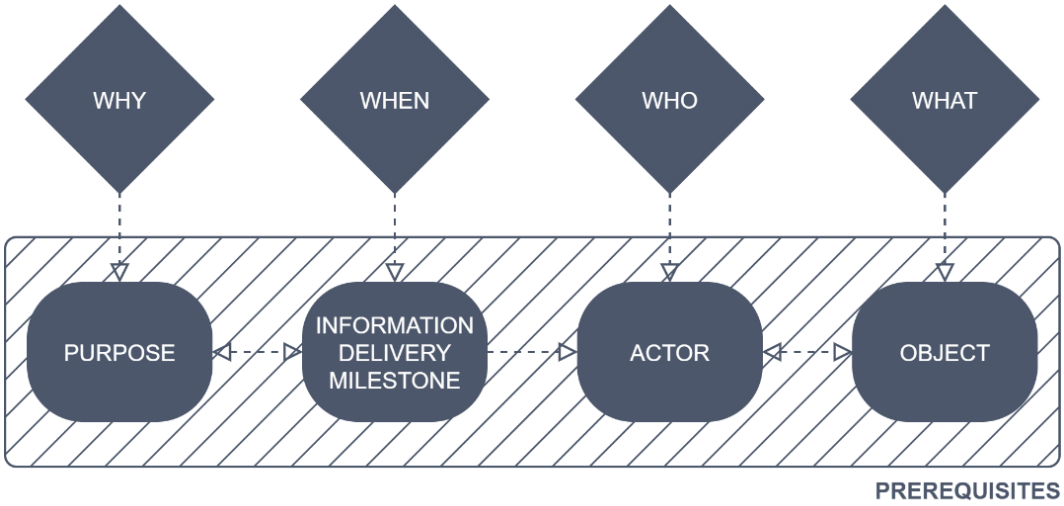


Figure 2 - Prerequisites of Level Information Need

Creating LOIN documentation involves aligning project deliverables with the actors' exchange requirements and mapping them to specific BIM objects and attributes. In structured environments, this process is supported by the *Information Delivery Manual* (IDM) methodology and can be represented digitally using data templates compliant with ISO 23387.

Despite the clarity offered by the LOIN framework, its implementation remains inconsistent across tools and projects. One common challenge is the manual interpretation of requirements (often formatted in spreadsheets or narrative texts), which leads to varying levels of understanding and execution. Moreover, current software solutions often do not integrate LOIN documentation authoring with semantic support or intelligent feedback, leaving non-expert and AECO professional users exposed to misinterpretations, therefore affecting efficiency and the development of BIM workflows.

In this context, LOIN not only defines the quantity and format of information but also offers a foundation for automation and validation. When combined with semantic standards and intelligent systems, it can enable dynamic generation of requirement-compliant content and streamlined information delivery across platforms.

2.3. Standards Governing BIM Information Requirements

The successful definition, validation, and exchange of information in BIM processes depend on a structured framework of international standards provided by ISO. These standards offer guidelines and models that ensure consistency, reliability, and interoperability across disciplines and software environments. The development of the prototype subject of the thesis is directly informed by several of these standards, which together define the semantics, format, and timing of data exchange throughout a project's lifecycle.

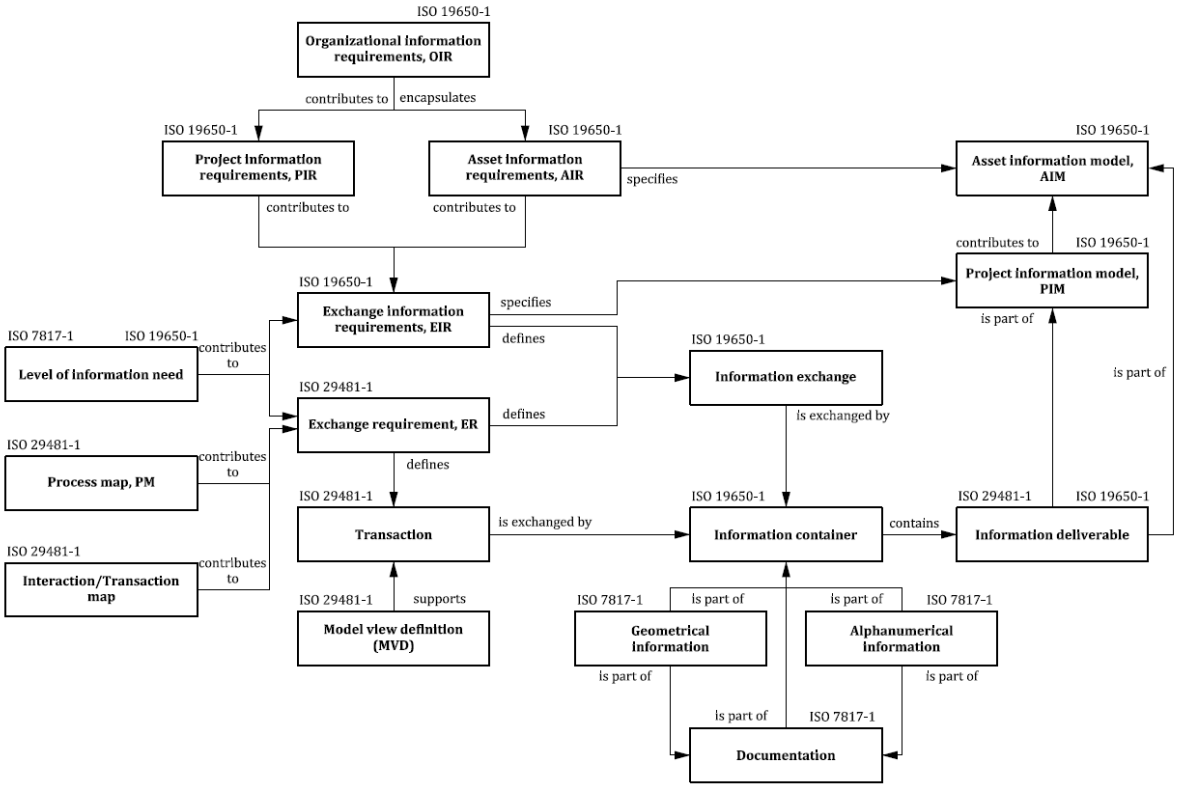


Figure 3 - Conceptual relationships between ISO 7817 and other ISO Standards (ISO7817-1, 2024)

2.3.1. ISO 19650 Series – Information Management Framework Across Delivery Stages

The ISO 19650 series defines the foundational principles for organizing and managing information throughout the lifecycle of a built asset within a collaborative BIM environment. By establishing standardized protocols for information planning, delivery, and validation, it ensures consistency across teams, systems, and project stages. This includes clear guidelines on the definition of information requirements, responsibility allocation, and formal procedures for data validation and approval (ISO19650-1, 2018).

One of the most significant contributions of the series lies in formalizing *when* and *why* information should be created, validated, and exchanged. Rather than leaving these decisions to ad-hoc interpretations, the ISO 19650 framework promotes a predictable and coordinated workflow aligned with project objectives. This principle is especially relevant when designing automated systems, where the quality of information guidance depends on accurate definitions of roles, phases, and data obligations.

At its core, the standard emphasizes the use of a *Common Data Environment* (CDE) to ensure that information remains accurate, trusted, and traceable. It encourages collaborative information production, structured task allocation, and formal information exchange based on well-defined requirements. These principles underpin the logic of this thesis prototype, particularly in interpreting task roles and timing for generating information (ISO19650-1, 2018). The project subject to this thesis has been designed to assist the user side in understanding responsibilities and delivery protocols as defined in the CDE framework, e.g. "who should provide the data, what data, and when it should be delivered", automating support for standardized information workflows (ISO19650-1, 2018).

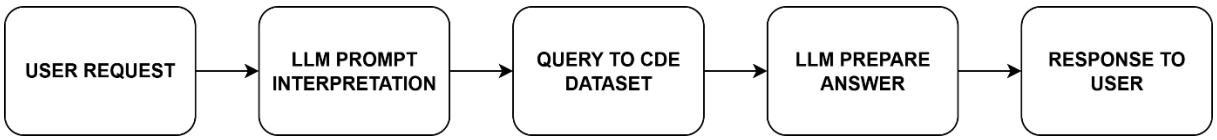


Figure 4 – LLM-Driven CDE definitions

Subsequent parts of the series have extended these principles to address later project phases and complementary concerns. ISO 19650-3 focuses on information management during the operational phase of assets, ensuring continuity of information use beyond design and construction. ISO 19650-4 provides a structured framework for information exchange, with an emphasis on machine readability and trustworthiness of data deliverables. Finally, ISO 19650-5 addresses security-minded information management, highlighting the need to safeguard sensitive information in digital collaboration environments. Together, these parts consolidate the ISO 19650 framework as a comprehensive reference for the entire asset lifecycle, aligning BIM processes with governance, security, and interoperability requirements.

2.3.2. ISO 29481-1 (IDM) – Describes processes for defining information exchange

The IDM, defined in the ISO 29481-1 standard, establishes a structured methodology for defining and modeling Exchange Information Requirements (EIR) across the lifecycle of a project. It clearly articulates who needs specific information, *what* information is needed, when it must be delivered, and in what format (ISO19650-1, 2018). This is especially relevant when defining LOIN, as IDM provides a procedural framework that ensures each data request or exchange is purposeful, role-specific, and aligned with project milestones.

By formalizing these exchange rules, IDM prevents both under-specification (missing data) and over-specification (irrelevant or excessive information), two issues frequently encountered/two in manual LOIN authoring. When combined with the LOIN framework, IDM enables a systematic mapping of responsibilities, deliverables, and data structures, reducing ambiguity in information flows. This

integration ensures that project information exchanges are not only consistent but also verifiable, supporting the delivery of accurate and fit-for-purpose data throughout the asset lifecycle.

As such, IDM functions as a bridge between project information requirements and their practical implementation in BIM processes. It operationalizes the high-level principles of the ISO 19650 series by defining structured workflows for information delivery, which can then be further supported by data templates compliant with ISO 23387. This layered approach strengthens interoperability, enabling reliable communication across disciplines, organizations, and digital platforms.

2.3.3. ISO 7817-1 – Levels of Information Need

The ISO 7817-1 standard introduces a formalized and internationally consolidated approach to specifying LOIN within BIM processes. ISO 7817-1 standard provides a structured framework for defining precisely what information is required, to what degree of granularity, and for what purpose. The standard organizes this specification into three interrelated dimensions: geometric information, alphanumeric properties, and documentation (ISO7817-1, 2024).

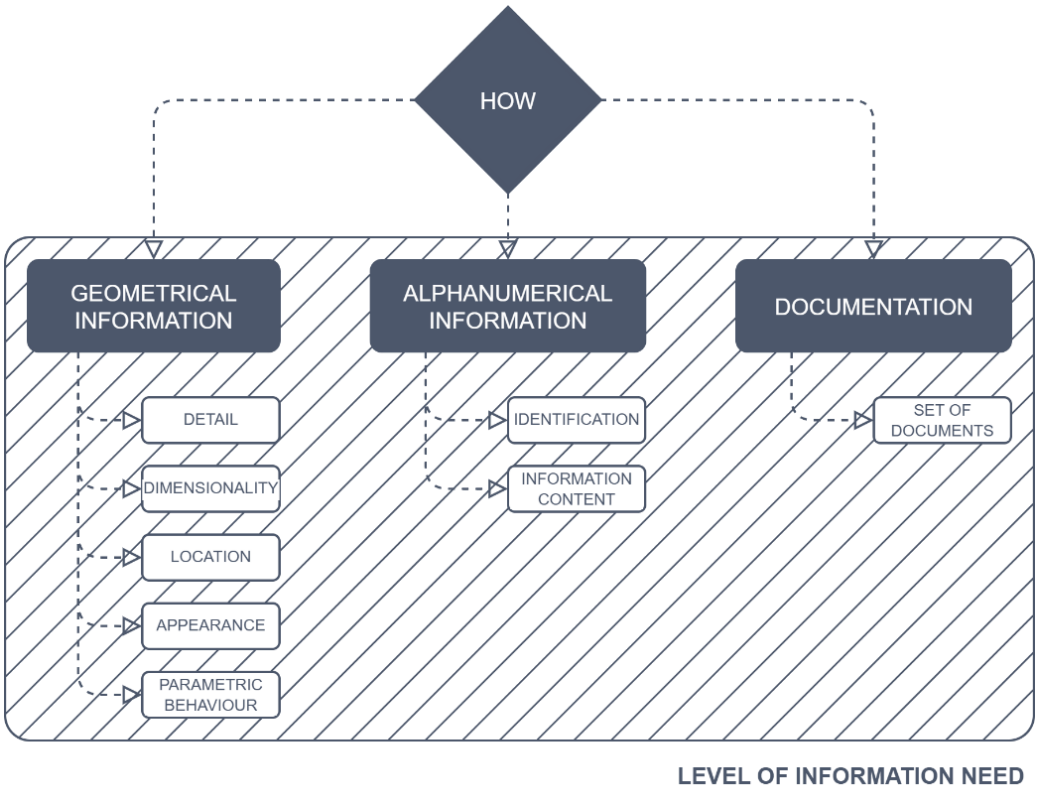


Figure 5 - Level Information Need

By structuring LOIN as a purpose-driven specification rather than a prescriptive checklist, the standard helps stakeholders avoid two recurrent pitfalls in BIM practice: *over-modeling* (where unnecessary detail inflates the model and complicates its use) and *under-modeling* (where essential data is missing, impairing decision-making). This purpose-orientation makes ISO 7817-1 particularly relevant for digital environments focused on generating, validating, or exchanging information requirements, as it ensures semantic alignment between user intent, data deliverables, and standardized outputs.

In the context of automation, ISO 7817-1 plays a critical role by introducing measurable parameters for assessing information completeness and appropriateness. Its structure presents an opportunity for the development of AI-driven systems that can evaluate the relevance of user queries, filter retrieved content accordingly, and produce outputs that are not only technically accurate but fit-for-purpose. These semantic constraints could improve the quality and consistency of generated documentation and enable the prototype to suggest specific property sets or modeling scopes based on project phase and actor role.

By replacing regional standards such as EN 17412-1, ISO 7817-1 consolidates the concept of LOIN at the international level, offering a unified reference point for the AECO sector. This transition enhances global interoperability, ensuring that requirements specified under the LOIN framework are consistent, verifiable, and reusable across projects, platforms, and jurisdictions.

2.3.4. Other Relevant Standards

Other relevant standards help to define the BIM framework boundaries, for example ISO23387 introduces a formal method for creating data templates for construction objects. These templates organize information about properties (alphanumeric attributes) in a machine-readable format, allowing for interoperability between software tools and ensuring that information requirements are consistently structured across the asset lifecycle (ISO23387, 2020). This standard plays a foundational role in the definition of construction data and supports the semantic clarity needed for digital workflows.

Closely related to these developments is the ISO7817-1 series, which has consolidated and expanded the concepts originally introduced in EN 17412-1. While EN 17412 standard first established a structured approach to LOIN, ISO 7817-1 elevates this framework to an international standard, ensuring global alignment and interoperability. The ISO7817-1 series provides a comprehensive and extended framework for the modeling of information requirements, offering both human- and machine-readable structures for defining, organizing, and validating LOIN.

By deriving from and ultimately replacing the earlier European standard, ISO7817 ensures continuity while establishing a unified reference point for the AECO sector. Its alignment with semantic structuring and automation objectives creates the foundation for rule-based processes that facilitate the reuse and verification of information across project phases, actors, and digital platforms.

2.4. Data Dictionaries (bSDD)

The *buildingSMART Data Dictionary* (bSDD) is an open, multilingual repository that provides standardized definitions for construction-related terms, classifications, and property sets. It serves as a reference point to ensure semantic consistency across software platforms and international projects. Terms such as *fire resistance* or *thermal conductivity* are encoded in a structured, machine-readable format, making them interoperable and reusable within BIM environments (buildingSMART, 2025).

The primary goal of bSDD and other domain-specific data dictionaries is to harmonize terminology and property definitions throughout the construction and asset management lifecycle. This semantic alignment is key for accurate data exchange and regulatory compliance, especially as BIM compliance evolves toward higher levels of automation and integration. Recent research studies how structured data

dictionaries support the transformation of unstructured catalogues into ontology-ready, standardized digital assets, forming the backbone of semantic BIM systems (Schilling & Clemen, 2024).

The incorporation of bSDD on AI agent development could leverage this resource to enrich the terminology and improve semantic consistency. By linking AI prompts to standardized term definitions in bSDD, the assistant could provide more accurate, interoperable documentation and suggest properties aligned with industry norms (Alexiev, et al., 2023).

2.5. Semantic Structuring in AECO: Ontologies and Data Dictionaries

As BIM workflows evolve towards greater automation and interoperability, the need for semantically structured information becomes increasingly critical. Semantic structuring enables machines to understand, validate, and manipulate building data in a consistent and meaningful way (Lee, et al., 2016; Teclaw, et al., 2023). In the AECO industry, this structuring is made possible through the use of ontologies and data dictionaries.

Ontologies are formal representations of domain knowledge that define the entities, attributes, and relationships relevant to a specific context. In the AECO industry, RDF-based ontologies have been developed to structure IFC-related concepts into machine-readable formats, enabling BIM data integration into broader knowledge graphs and linked data environments (Oraskari, et al., 2024; Alexiev, et al., 2023). In parallel, a standard-based ontology network specifically designed for information requirements formalizes how these requirements can be represented and linked to standardized properties, supporting consistent interpretation across stakeholders and tools (Mellenthin Filardo, et al., 2024).

By defining building elements like walls, doors, or HVAC systems as classes with specific properties and hierarchical relationships, ontologies support machine reasoning, validation, and automated querying (Oraskari, et al., 2024; Katranuschkov, et al., 2003). For example, a system can infer that a fire-rated partition is a subclass of a wall and that it must include certain performance properties in compliance scenarios. This semantic layering is particularly relevant for ensuring LOIN completeness based on standardized rules, and it aligns with the goal of encoding information requirements in a form that is interpretable and checkable by machines (Mellenthin-Filardo et al., 2024).

In parallel, “data dictionaries” such as the bSDD, as described in the previous point, serve as centralized repositories of construction- related terms, classifications, and property sets. While not ontologies in the strictest sense, they fulfill a complementary role by providing harmonized definitions and structured vocabularies. For instance, bSDD supports multilingual interoperability and links terms to external standards such as ISO 23386 and ISO 23387 (Alexiev, et al., 2023; Oraskari, 2021).

When used together, ontologies and data dictionaries could offer a robust foundation for both defining and validating BIM content. They enable applications to ground AI outputs in formally defined structures, reduce ambiguity in terminology, and improve documentation consistency (Teclaw, et al., 2023). In this thesis, the standard-based ontology network for information requirements developed by Mellenthin-Filardo et al. (2024) will be adopted as a reference schema to represent LOIN-aligned

information requirements, aiming to connect requirement classes and properties with standardized terminologies (e.g., via data dictionaries), and to support graph-based validation patterns where feasible.

Nevertheless, semantic technologies in AECO remain underutilized. Current tools often lack intuitive interfaces or require specialist knowledge in knowledge engineering or SPARQL querying. As a result, their adoption is typically limited to research or high-complexity environments. One of the aims of this thesis is to explore whether AI-driven interfaces (specifically those powered by Large Language Models) can act as semantic mediators, bridging this knowledge gap.

2.6. Artificial Intelligence in AECO: LLMs and RAG Pipelines

AI has emerged as a transformative technology within the AECO sector. With the growing demand for structured and timely information, AI-based systems are increasingly considered as a means to support the organization, validation, and interpretation of data within BIM workflows (Du, et al., 2024). One of the most promising developments in this context is the rise of LLMs, which can understand and generate human-like text based on extensive training over diverse datasets.

Recent research has explored how these models can be enhanced through techniques such as RAG, which allows external knowledge sources to be introduced into the response generation process (Lewis, et al., 2020). This architecture offers a way to combine pretrained language fluency with dynamic retrieval of structured content such as standards, domain ontologies, and technical guidance. Its relevance in BIM lies in its potential to mediate complex terminology and connect user queries to semantically meaningful results.

The integration of LLMs with formal construction knowledge remains a relatively new field, but several studies have outlined both its promise and its limitations (Höltgen, et al., 2025). Challenges such as aligning free text outputs with standard-compliant formats, preserving semantic precision, and adapting models to highly specialized vocabularies are active areas of research. Moreover, questions around transparency, model governance, and interoperability remain central to current discourse.

2.6.1. Overview of Mainstream LLM Architectures

Recent developments in AI have led to the emergence of a wide range of commercial LLMs, such as OpenAI's GPT family, DeepSeek, Google's Gemini, and many others. These models have demonstrated a high level of proficiency in interpreting and generating natural language interactions. Trained on extensive multilingual and multi-domain text corpora, they are capable of interpret (Höltgen, et al., 2025; Funk, et al., 2023).

In the context of the AECO sector, these capabilities are particularly relevant for tasks that involve interpreting technical documentation, understanding regulatory standards, and supporting the creation of structured content. The ability of LLMs to process and articulate *domain-specific* context positions them as valuable tools for improving information-driven processes in BIM environments (Du, et al., 2024).

These models can be deployed in several different configurations, each offering particular advantages and limitations. Cloud-based *Application Programming Interfaces* (APIs) provide scalable access to the

most up-to-date model versions; however, they may present challenges related to internet dependency and data governance. Locally hosted models, although limited in training scope and processing capacity, provide greater control over data and support offline usage. Custom endpoints serve as an intermediate option, allowing organizations to tailor a model's behavior through controlled input and output parameters. While they offer flexibility and partial adaptability, their implementation typically demands additional infrastructure and technical expertise, especially when adjusting the model's performance for specific use cases (Höltgen, et al., 2025; Du, et al., 2024).

2.6.2. Retrieval-Augmented Generation (RAG)

RAG is a framework that integrates LLMs with external information retrieval systems. Unlike traditional LLMs, which rely solely on static pretraining data, RAG pipelines enable dynamic access to up-to-date or domain-specific content (Lewis, et al., 2020). This is particularly relevant in the context of BIM, where accurate responses often require referencing normative documents, such as ISO standards, or project-specific repositories.

The typical RAG architecture consists of several sequential components. *Chunking* refers to dividing input documents into discrete segments, facilitating manageable and relevant retrieval units. These chunks are then transformed into vector representations through *embedding*, allowing them to be stored in vector databases for efficient retrieval based on similarity. At query time, the system retrieves the most relevant segments and integrates them into the prompt provided to the LLM, enabling the generation of responses grounded in contextually accurate information (Lewis, et al., 2020).

Recent extensions to this architecture include approaches like *Graph-RAG,* which incorporate knowledge graphs into the retrieval process. By structuring content through semantic relationships and ontological links, this method enhances the contextual understanding and traceability of responses (Xu, et al., 2024). Such structured *indexing* can help mitigate hallucinations and ensure a more transparent mapping between input queries and retrieved information, an essential consideration in information-sensitive domains such as BIM documentation and LOIN specification.

2.6.3. Prompt Engineering, Chunking & Indexing Strategies

Prompt engineering refers to the structured design of user inputs that elicit accurate, relevant, and context-aware responses from LLMs In BIM workflows, this involves referencing standards, object properties, or specific information needs in clear and organized language, ensuring that the model understands the task and produces compliant outputs (Funk, et al., 2023).

An emerging approach within this field is agent-based refinement, in which AI autonomously revises its own prompts to clarify vague user queries or follow up with more targeted sub-questions. This technique reduces the cognitive burden on users and makes the interaction more intuitive, particularly for non-experts. It also improves alignment between the user's intent and the system's response.

In RAG systems, the effectiveness of the response depends heavily on how documents are segmented and stored. This process, known as chunking, breaks source documents into smaller, manageable units to optimize retrieval precision. Parameters such as chunk size, degree of overlap, and semantic structure

play a decisive role in the accuracy and contextual relevance of retrieved information (Lewis, et al., 2020).

Indexing strategies complement this process by enabling fast, context-aware search operations across vector databases. Recent research has shown that semantically guided chunking, where document segments are aligned with domain-specific sections (e.g., standard clauses or classification systems), enhances both retrieval accuracy and semantic coherence (Lewis, et al., 2020; Xu, et al., 2024).

Additionally, protocols like the *Model Context Protocol* (MCP) propose structured methods to relate modeling context with information requirements. This emerging specification could offer a robust foundation for enhancing indexing logic, enabling AI to match user queries to relevant regulatory, technical, or project-specific contexts (Hou, et al., 2025).

2.6.4. Deployment Trade-offs a comparative between the functionalities of Cloud APIs, Custom Endpoints and Local Models Solutions

As language models increasingly support domain-specific tasks, *prompt engineering* has emerged as a key practice for optimizing their performance. It involves designing input prompts that clearly convey the user's intent while aligning with the model's structure of understanding. In regulated domains such as BIM, this may include referencing specific clauses from standards, naming conventions, or detailed object attributes. The clarity and specificity of the prompt can significantly influence the quality, accuracy, and reproducibility of the AI-generated output.

Beyond static prompts, recent research explores dynamic or agent-guided prompt refinement. In this approach, the AI system can iteratively reformulate unclear inputs, ask clarification questions, or generate sub-queries to narrow down the information needed. This method offers a promising direction for increasing accessibility, especially for users with limited familiarity with technical standards or documentation protocols (Xu, et al., 2024).

Complementing prompt design, chunking and indexing strategies play a crucial role in retrieval-based systems. Choices around chunk size, overlap, and segmentation logic affect the precision and completeness of retrieval results. Indexing structures, including traditional vector search and ontology-aware graphs, determine how these chunks are stored and retrieved in response to user queries. Together, these strategies shape the AI's ability to locate, interpret, and recontextualize content for output generation.

Current literature highlights that the balance between chunk granularity and retrieval scope remains an open challenge, especially in fields that rely on hierarchical or cross-referenced documentation such as BIM (Xu, et al., 2024). The combination of effective prompt design and refined indexing may be key to enabling scalable and reliable AI support in digital construction workflows.

A promising direction in this domain is the integration of structured input protocols such as the MCP. Developed as a standard to contextualize queries in BIM environments, MCP provides a structured vocabulary and grammar for asking precise, context-aware questions related to model objects, actors, and lifecycle stages. Its use could standardize how prompts are formulated for LLMs, improving both interpretability and semantic alignment.

2.7. Conceptual Integration: Opportunities and Research Gaps

The previous sections have examined the essential building blocks that support structured information management in BIM workflows, including regulatory frameworks, semantic modeling strategies, and AI tools. Collectively, these components form a robust theoretical basis for enhancing automation and reliability in data delivery processes across the AECO sector. However, the integration of these resources into accessible, intelligent authoring environments remains an open challenge.

Despite the detailed guidance offered by ISO 19650, ISO 7817-1, ISO 29481-1, and other related standards, the tools available to practitioners are often limited in scope and usability. Most platforms rely on static forms or rigid templates that require users to know the concepts and semantics to manually encode requirements without dynamic feedback or validation. As a result, these systems frequently overlook the user context, leaving practitioners, especially those without specialized training, struggling to interpret standards and deliver consistent documentation. This disconnect hampers the broader implementation of LOIN methodologies in everyday workflows.

Semantic technologies, including ontologies and standardized data dictionaries like bSDD, have been proposed as a way to structure and validate domain-specific information. While conceptually promising, their application is still constrained by technical barriers. Ontologies often require specialized knowledge to create, query, and integrate into existing software environments. In addition, most implementations do not support automatic reasoning over semantic rules or real-time linkage to external dictionaries. As a result, the benefits of semantic alignment, such as automated classification, improved data quality, or logic-based validation remain underused in practical applications.

Meanwhile, developments in *Natural Language Processing* (NLP), particularly the emergence of LLMs and RAG pipelines, offer new ways to make technical content more accessible. These systems could interpret unstructured input, retrieve relevant information, and generate structured outputs in response to user prompts. However, their application in AECO, specifically in the context of standard-driven documentation generation, is still in early stages. The use of LLMs in this domain has yet to be systematically studied, particularly with respect to compliance with the existing ISO framework.

The intersection of these three domains, semantic technologies, information management standards, and AI-based assistance, represents a timely and underexplored research frontier. No existing solution fully integrates RAG-enabled AI, ontology-based reasoning, and standardized information requirement models in a form that is both practical and intelligible for real users.

This thesis seeks to address that gap by conceptualizing and prototyping a methodology for AI-assisted LOIN documentation. The proposed framework draws upon international standards, applies semantic structuring principles, and leverages retrieval-enhanced language models to enable assistance sensitive to the context. In doing so, the research contributes both to the practical implementation of LOIN and to the broader goal of aligning intelligent systems with the information needs of BIM workflows.

This page is intentionally left blank

3. PROTOTYPE METHODOLOGICAL DEVELOPMENT BIM/LOIN ASSISTANT TOOL

The present chapter documents the end-to-end design and refinement of the assistant, from initial RAG tests to advanced indexing and autonomy controls.

3.1. Research Approach and Design

The methodological framework underpinning the development of the prototype is primarily qualitative and exploratory. Rather than testing a fixed hypothesis in a controlled environment, the project adopts a flexible and iterative approach. It seeks to examine the potential and limitations of current AI technologies, particularly LLMs and RAG strategies, for automating semantically accurate and standards-aligned LOIN interactions.

The research process follows a trial-and-error methodology, characterized by progressive learning and adaptation. Initial experiments involved the direct interaction with commercial LLMs to evaluate their raw capability to interpret BIM concepts and queries related to LOIN. Based on the limitations observed, the process evolved into the design of custom pipelines using Python, integrating document retrieval, semantic chunking, vector databases, and later external ontological references.

Although the development is not formally aligned with a single software engineering methodology, the process reflects characteristics of both iterative prototyping and design-based research. In practice, solutions are continuously refined based on observed limitations, emerging opportunities, and the formal requirements defined by industry standards. In this sense, the prototype is treated as a research artifact, developed and tested as part of a broader investigation into the feasibility of intelligent documentation generation in BIM workflows.

The evaluation strategy does not currently include user testing, as the prototype is still under development. Instead, the research focuses on standards-based validation, examining the system's ability to respond to prompts in alignment with ISO 19650-1 and -2 (information management and delivery framework), ISO 29481-1 (IDM), ISO 23387 (data templates for construction objects), and ISO 7817-1 (LOIN). These references also form the foundation for the knowledge embedded within the system's RAG pipeline.

This hybrid methodology combines conceptual exploration, technical implementation, and standards-based assessment aiming to deliver both practical insights into AI capabilities and a critical reflection on their integration within regulated BIM environments.

3.1.1. Methodology Overview

Automating the definition of LOIN in BIM workflows requires us to compare the strengths of general-purpose LLMs (such as ChatGPT and Ollama) against a custom solution. To do this, we adopt a two-phase evaluation.

The first phase is the Diagnostic Phase. We submit a fixed set of prompts and a guided conversational script to each mainstream LLM in its default configuration. In this way, we observe how well these models retrieve, interpret, and organize LOIN content without any added domain knowledge or retrieval augmentation.

In the second phase, which we call the Guided Phase, we provide exactly the same prompts and conversational flow to our custom solution. This phase combines a retrieval-augmented backend with formal ontologies that have been aligned to ISO standard. By comparing the outputs of the two phases, we can attribute the gains in definition accuracy, completeness of information, and user guidance directly to the inclusion of structured semantic context, as supported by recent studies that apply LLMs for ontology construction (Funk, et al., 2023) and for semantic enrichment of infrastructure data (Höltgen, et al., 2025).

Each of these phase's maps to one of our core research objectives. The Diagnostic Phase measures the baseline performance of unmodified LLMs. The Guided Phase verifies the added value of ontology-driven processing. Comparing results across both phases allows us to quantify overall improvements using a set of standardized metrics.

All of the model responses are scored on four criteria: accuracy, completeness, structure, and interaction quality. To enhance reliability, the researcher will first calibrate the rubric by scoring a pilot set of sample responses and refining any ambiguous criteria. Discrepancies between rubric definitions and observed scoring patterns will be resolved through iterative rubric adjustment. Once finalized, the rubric will be applied consistently across all model outputs.

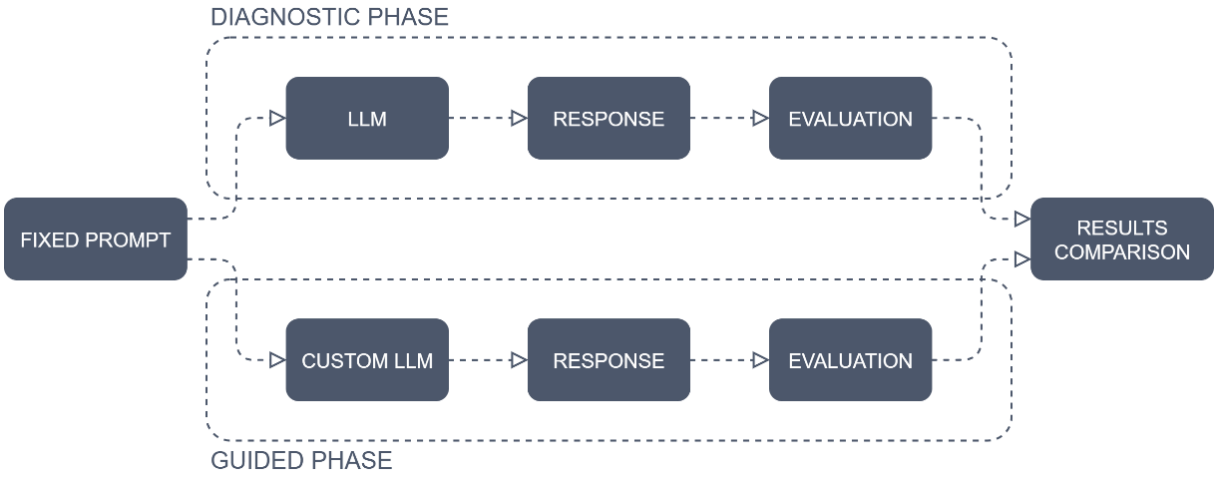


Figure 6 - Two phase evaluation workflow

3.1.2. Input Variables to Test

A small set of minimal, repeatable prompts ensures that each model faces an identical test. By limiting the evaluation to two core questions, we eliminate confounding variables such as prompt length or complexity. This approach isolates the model's true semantic grasp of LOIN concepts, providing a fair basis for cross-model comparison.

Each prompt directly reflects requirements drawn from ISO 7817-1 and ISO 19650. The first question asks for a formal definition of LOIN, anchoring the response in the standard's conceptual framework. The second prompt probes awareness of the three LOIN dimensions (geometry, properties, and documentation) thereby testing a deeper, structural understanding.

ID	Prompt Input	Interaction Type	Goal
Q1	*"What is the Level of Information Need in BIM?"*	Fixed	Tests basic LOIN definition alignment with ISO 7817-1.
Q2	*"Do I need geometry, properties, or documentation for lighting?"*	Fixed	Assesses if the model understands the 3-part LOIN structure.

Table 1 - Formal Definition Query

When presented to general-purpose LLMs, we expect fluent but sometimes imprecise definitions. Models may approximate the notion of LOIN but omit explicit reference to alphanumeric versus geometrical information or fail to mention documentation requirements. Such baseline behavior highlights areas where semantic grounding is weak.

3.1.3. Conversational Flow Simulation

A realistic dialogue with a non-expert AECO user aims to reveal how well an AI assistant can scaffold understanding. Rather than isolated questions, this simulation matches the stepwise logic a practitioner might follow when defining LOIN.

We sequence the interaction to mirror required inputs: first clarifying purpose, then identifying the project milestone, specifying the relevant elements, generating the LOIN table and finally accommodating revisions. This order mimics the logical progression set out in ISO 7817-1 and supports testing each knowledge transition point.

Step	Prompt Input	Roleplay Context	Goal
C1	*"I need to define the information for some elements of my project."*	Unclear goal	Triggers model to ask about purpose/milestone.
C2	*"I'm working on a house design, but I don't know what data I should include."*	Scenario disclosed	Model should infer or explain the need for purpose/milestone.
C3	*"We are currently developing the plans for the contractor."*	User has approximate milestone idea	Model interprets likely project phase (e.g., design, early planning).
C4	*"What does 'purpose' mean here? What should I choose?"*	Lacks standards vocabulary	Model must explain purpose in simple terms and offer examples (e.g., cost estimation, compliance, coordination).
C5	*"Okay, let's say it's for cost estimation."*	User selects purpose	Satisfies ISO 7817-1 input requirement.

C6	*"Which elements should I define information for?"*	Needs scope guidance	Model suggests typical elements for renovation projects (walls, doors, HVAC, etc.).
C7	*"Let's go with walls, windows, and lights."*	User defines objects	Begins defining LOIN content.
C8	*"Can you give me a table with the information I need?"*	Requests structured result	Model outputs LOIN table (with geometry, properties, documentation per ISO).
C9	*"Oh, also include doors."*	Scope extension	Model updates table dynamically.

Table 2 - Conversational Simulation Query

Throughout this flow, we monitor whether the model asks clarifying questions, introduces standard terminology at the right moments, and adapts the output when the user extends scope. These cues indicate successful learning and flexible table generation.

3.1.4. Evaluation Metrics

Response quality is assessed along six macro metrics. LOIN Compliance measures how closely the output follows the structure and prescriptions of ISO 7817-1 and ISO 19650-1/2. Semantic/Ontology Compliance evaluates the consistency of terminology, templates, and methodology with the referenced standards, including ISO 23387 and ISO 29481-1. Completeness determines whether the response covered the user request with sufficient detail and the concepts defined in ISO 7817-1 are covered with sufficient detail. Precision verifies the correctness of terminology and the consistency through the session, minimizing contradictions and hallucinations. Traceability examines whether requirements and decisions are justified and auditable through explicit links to rules, sources, or rationale. Structure/Legibility assesses the clarity, organization, and usability of the delivered response.

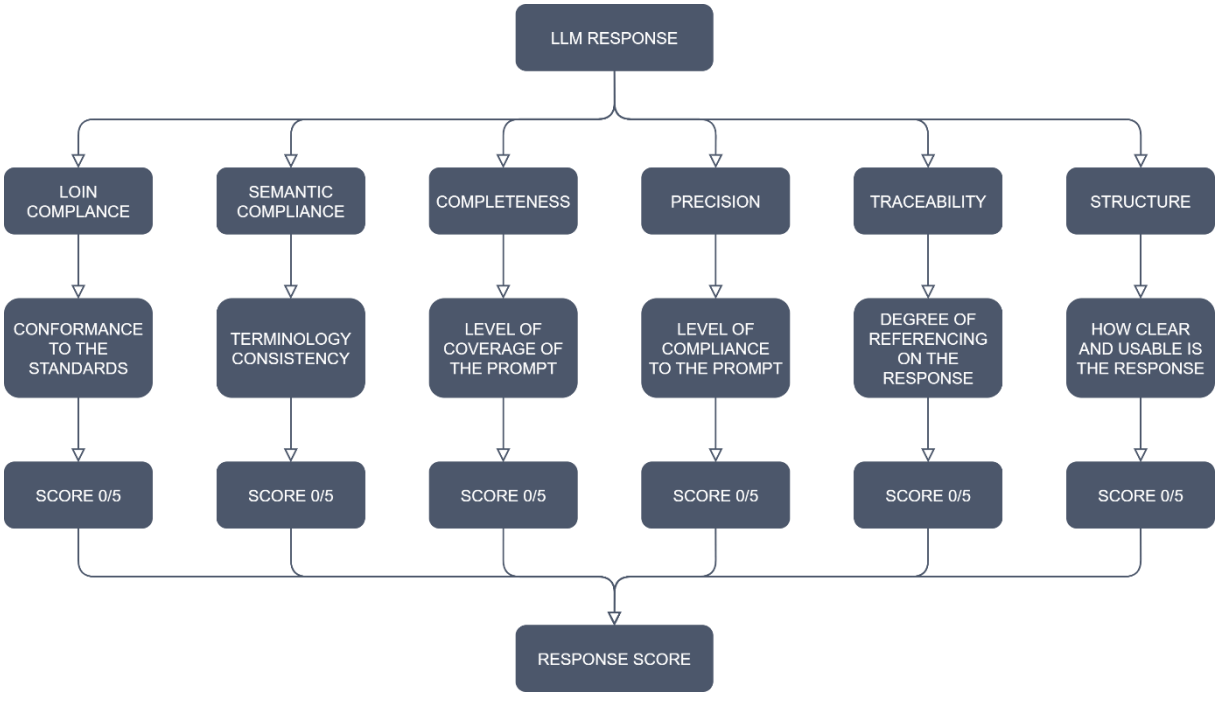


Figure 7 - Evaluation Diagram

Each model output is scored on a five-point scale for every metric, where zero denotes complete failure and five represents full compliance with no omissions. At the same time each model will be run three time for each of the sessions in order to assess consistency in the results.

Once scoring is complete, the average values are computed for each metric and perform mean and statistical analysis to determine whether differences between the stand alone LLMs against the custom solutions are significant. These analyses directly inform our conclusions on the quantitative value added by structured semantic integration.

3.2. Development Roadmap

This subsection presents the prototype's progression through a series of stable versions that consolidate design choices, exploring the characteristics and limitations of each of these iterations.

3.2.1. Phase 1:

This phase established a feasibility baseline for a standards-aware assistant, but results from this exploratory iteration were inconclusive. The intent was to structure single-turn guidance using the LOIN framework of ISO 7817-1. When reviewed against check criteria, typical outputs failed completeness and consistency and did not achieve dependable interaction or performance at decision points. Phase 1 is therefore recorded as a learning stage that motivates the redesign in Phase 2 rather than claiming functional adequacy.

3.2.1.1. Stable Version 1.1:

The stable version 1.1 of the prototype (Appendix 1, v0.0.6) was streamlined to focus on rapid ingestion of the LOIN ontology and core ISO standards, with minimal user interface overhead. At startup, the system loads the LOIN ontology (in TTL format) and retrieves all class labels for valid fields that guide data collection. It then extracts text from the key PDF standards (ISO 19650-1, ISO 19650-2, and ISO 7817-1), concatenates ontology labels and standard excerpts into a single corpus, and prepares it for embedding. This iteration preserves a single-step conversational style aiming for a lean and clear mapping between user inputs and semantic classes.

The processing pipeline in stable version 1.1 proceeds as follows:

1. Ontology Loading: load_ontology_graph reads the TTL file, and get_all_classes extracts class URIs and labels into valid_fields.
2. Standards Extraction: extract_text_from_pdf is applied sequentially to each ISO/EN PDF, yielding a unified text block.
3. Document Preparation: split_into_docs chunks the combined ontology and standard text into passages suitable for embedding.
4. Embedding & Indexing: OllamaEmbeddings generates vector representations, stored via Chroma.from_documents.
5. QA Chain Construction: RetrievalQA.from_chain_type pairs the Chroma retriever with an Ollama LLM endpoint to form a RAG pipeline.

6. Conversational Input Collection: collect_loin_inputs iterates through an ordered list of LOIN fields—prompting the user via llm.invoke and capturing answers in sequence.
7. Prompt Generation: generate_loin_table_prompt formats the collected inputs into a final instruction that instructs the model to emit a markdown table compliant with BS EN 17412-1 and ISO 19650.
8. Execution & Save: The RAG chain runs the table prompt and save_markdown writes and opens the resulting file in the docs/ folder.

Despite these, several key limitations remain from the starting point. There is no validation of user entries against the ontology, so free-text answers may fall outside the controlled vocabulary. All fields are mandatory and sequential, preventing users from skipping irrelevant items, as a consequence the prototype lacks of real conversational capacities. The pipeline still lacks error handling (e.g., for missing PDF files or injection of non-standard terms), and the single-threaded *Command-Line Interface* (CLI) offers no real-time feedback or dynamic hinting.

Table 3 shows the Libraries and Modules used in this prototype aiming identify the limitations on their implementation.

Library / Module	**Role**	**Limitation**
os	Construct file paths and handle filesystem operations.	No explicit checks for path existence or permissions.
fitz (PyMuPDF)	Open and extract raw text from PDF standards.	Drops document structure (headings, tables); fails on scanned or image-only PDFs.
webbrowser	Launch the generated markdown file in the user's default browser.	No control over browser errors or user prompts; blocking call.
rdflib.Graph	Parse the TTL ontology file into an RDF graph.	Does not enforce ontology consistency; overlapping classes are not merged.
langchain_community. llms.Ollama	Send prompts and context to a locally hosted Ollama LLM.	Tight coupling to Ollama endpoint; no fallback if model is unavailable.
langchain_community. vectorstores. Chroma	Index and retrieve vector embeddings for RAG.	In-memory only; not persisted across sessions; limited scalability.
langchain_community. embeddings.Ollama Embeddings	Generate vector embeddings from text passages.	Embedding quality depends on model version; no batching controls.
langchain.chains. RetrievalQA	Build a RAG chain combining retriever and LLM.	Single-pass retrieval; no multi-hop reasoning or reranking.
langchain.text_splitter. Character TextSplitter	Divide large text into fixed-length chunks suitable for embedding.	Fixed chunk size may split semantic units; no overlap tuning.

langchain.docstore. document. Document	Wrap each text chunk in a Document object with metadata.	Minimal metadata support; does not capture ontology relationships.

Table 3 - Libraries and limitations v1.1.

3.2.1.2. Stable Version 1.2:

In this phase, the prototype (Appendix 1, v0.1.3) refines its terminology mapping by loading the standards-based ontology network from Hagedorn & Liu, 2023 and extracting all class labels into a controlled vocabulary. Users are prompted, in sequence, to supply each LOIN field. Their answers are compared against the valid vocabulary list by way of fuzzy matching and WordNet lemmatization. Valid inputs are normalized to the canonical terms, and any unmatched response triggers a repeat question. The system then formats the collected information into a table-generation prompt that the OpenAI Chat model should use to produce a markdown table compliant with ISO 7817-1 and ISO 19650.

The processing pipeline executes in these steps:

1. Ontology loading and label extraction: The function load_ontology_graph parses the TTL file and get_all_classes extracts all class URIs and rdfs labels into valid_fields.
2. Standards ingestion and text chunking: extract_text_from_pdf reads each PDF and combines the raw text with ontology documentation, then CharacterTextSplitter breaks the combined text into passages of about one thousand characters.
3. Embedding and vector store creation: OpenAIEmbeddings converts each passage into a vector and Chroma.from_documents indexes those vectors in memory.
4. RAG chain construction: RetrievalQA.from_chain_type pairs the Chroma retriever with the ChatOpenAI model to form a RAG pipeline.
5. Conversational input collection: collect_loin_inputs prompts the user for each valid field in sequence and applies rapidfuzz and WordNetLemmatizer to match responses to valid_fields.
6. Prompt assembly for table generation: generate_loin_table_prompt formats all validated inputs into a single instruction that asks the model to render a markdown table compliant with ISO 7817-1 and ISO 19650.
7. Execution and result saving: The RAG chain is invoked with the assembled prompt and save_markdown writes the output under docs / and opens it in the default browser.

Despite these enhancements, several limitations persist. The ontology is treated only as source text and never queried as a graph, so hierarchical relationships and property restrictions remain unused. Users must enter terms that appear exactly or very closely in the valid fields list; the system cannot suggest missing concepts or teach definitions. The conversational loop follows a fixed sequence with no ability to skip fields or branch based on context. Error handling is minimal, missing files or network errors halt execution.

Table 4 shows the Libraries and Modules used in this prototype aiming identify the limitations on their implementation.

Library / Module	Role	Limitation
os	Construct file paths and manage filesystem operations.	Missing explicit checks for file existence or permissions.
fitz (PyMuPDF)	Open and extract raw text from PDF standard documents.	Loses original formatting (headings, tables); fails on scanned or image-only PDFs.
rdflib.Graph	Parse the TTL ontology file into an RDF graph.	Only used to extract flat labels; no traversal of class hierarchy or property relationships.
langchain.text_splitter. CharacterTextSplitter	Split combined ontology and standards text into fixed-size passages for embedding.	Fixed chunk length can split semantic units arbitrarily; no overlap tuning.
langchain.docstore.document .Document	Wrap each text passage in a structured Document object for downstream processing.	Metadata limited to text content; does not capture ontology context.
langchain_community. embeddings.Open AIEmbeddings	Generate vector embeddings for each document chunk using OpenAI.	Dependent on external API availability; embedding quality varies with model configuration.
langchain_community. vectorstores.Chroma	Index embeddings in memory and support similarity searches.	In-memory only; lacks persistence and scalability for large document sets.
langchain.chains. RetrievalQA	Build a RAG chain combining the retriever and the LLM.	Single-pass retrieval; no support for multi-hop reasoning or reranking.
langchain_openai. ChatOpenAI	Send prompts and context to the OpenAI model and receive completions.	Fixed model and parameter settings; does not retain conversational state across calls.
nltk.corpus.wordnet	Provide WordNet lexical database for synonym lookup.	Limited to WordNet's English lexicon; no domain-specific term coverage.
nltk.stem.WordNet Lemmatizer	Normalize user input to base forms to improve matching accuracy.	Handles only basic English morphology; misses domain-specific variations.
rapidfuzz.process	Perform fuzzy string matching between user input and the controlled vocabulary.	Relies solely on string similarity; cannot infer semantic relationships or suggest corrections.
webbrowser	Open the generated markdown or HTML output in the user's default web browser.	No control over browser launching behavior; does not handle failures or alternative viewers.

Table 4 - Libraries and Limitations v1.2

3.2.1.3. Stable Version 1.3:

On Stable Version 1.3, the prototype (Appendix 1, v0.1.9) moves beyond a fixed prompt loop by first parsing a free-form user description into structured LOIN fields. It loads the full LOIN ontology in TTL format with rdflib and extracts every rdfs:label into a list of labels. At the same time, it fetches additional terms from the bSDD API for the construction domain and merges those into the suggestions pool. The system then reads each PDF standard with PyMuPDF, concatenates the text with the ontology documentation, and splits the result into manageable passages. All passages become embeddings in a Chroma vector store via OpenAIEmbeddings. A RetrievalQA chain powered by ChatOpenAI handles RAG.

Compared to version 0.1.3, the improvements include dynamic parsing of user scenarios, integration of bSDD terms, nested handling of complex fields such as geometry, alphanumerical and documentation subproperties, and a placeholder for SHACL-based ontology validation.

The processing pipeline now follows these stages:

1. Ontology Loading and bSDD Fetch: The TTL ontology file is parsed and all rdfs:label values are collected. The function fetch_bsd_terms calls the bSDD REST API to retrieve concept names for the construction domain. Labels and bSDD terms are combined into a suggestions map.
2. Standards Extraction and Document Preparation: Each PDF standard (ISO 19650-1, ISO 19650-2, ISO 7817-1, ISO 29481-1, ISO 23387) is opened with PyMuPDF. Text is extracted and concatenated with ontology documentation. CharacterTextSplitter chops the text into passages of roughly one thousand characters, each wrapped in a LangChain Document.
3. Embedding and Indexing: The OpenAIEmbeddings module converts every passage into a vector. Chroma.from_documents builds an in-memory vector store for semantic retrieval.
4. Initial Scenario Parsing: The function parse_user_input sends the user's free-form scenario description to ChatOpenAI with a prompt that asks for a Python dict containing the fields milestone, purpose, actor, object, geometry, alphanumerical information, and documentation. The returned dict seeds the context.
5. Conversational Field Filling: The function conversational_fill iterates over any missing fields. For simple fields it asks one question per field and offers suggested terms drawn from the suggestions map. For complex fields it loops over each subproperty and generates tailored prompts. All user answers are matched to suggestions via rapidfuzz and WordNetLemmatizer.
6. Ontology Validation: After the context is complete the function validate_with_shacl is invoked. In this version it returns True unconditionally, acting as a placeholder for future SHACL checks.
7. Table Generation and Save: The function generate_loin_table formats the context into a markdown table. save_markdown writes the file to the doc's folder and opens it in the default browser.

Despite the changes applied in this phase, the limitations persist. The ontology and bSDD network remain flat label lists so hierarchical relationships and property constraints are unused. The SHACL validation stub provides no real check. The field prompt sequence is still fixed with no branching or skip logic. Missing files network errors and invalid JSON from bSDD cause script termination.

Table 5 shows the Libraries and Modules used in this prototype aiming identify the limitations on their implementation.

Library / Module	**Role**	**Limitation**
os	Construct file paths and access environment variables.	No checks for path existence or permissions; environment errors cause silent failures.
fitz (PyMuPDF)	Open and extract text from PDF standard documents.	Strips formatting and structure; fails on scanned or image-only PDFs.
webbrowser	Open the generated markdown documentation in the user's default browser.	No feedback on errors or alternative viewers; blocks execution until browser closes.
requests	Fetch concept terms from the buildingSMART Data Dictionary REST API.	No retry logic or caching; network failures halt execution.
rdflib.Graph / RDFS	Parse the TTL ontology and extract rdfs:label values for valid field names.	Only flat label extraction; ignores class hierarchies, property restrictions, and relations.
langchain_community. vectorstores.Chroma	Load and query an in-memory vector index for similarity search.	In-memory only; not persisted across runs; limited scalability.
langchain_community. embeddings.OpenAI Embeddings	Generate vector embeddings for text chunks via OpenAI API.	Dependent on API availability and rate limits; embedding fidelity varies with model version.
langchain.chains.RetrievalQA	Combine retriever and LLM into a RAG chain.	Single-pass retrieval; no multi-hop reasoning or custom reranking.
langchain.text_splitter. CharacterTextSplitter	Split large text into fixed-size passages for embedding.	Fixed chunk size can split semantic units arbitrarily; no overlap control.
langchain.docstore. document.Document	Wrap each text chunk in a Document object with minimal metadata.	Does not capture ontology context or relationships.
langchain_openai. ChatOpenAI	Send prompts and context to the GPT-based model and receive responses.	Static model configuration; no stateful memory or dynamic context updating.
rapidfuzz.process	Perform fuzzy string matching between user input and controlled vocabulary.	Relies solely on string similarity; cannot infer semantic relationships or suggest corrections.

nltk.stem.WordNet Lemmatizer	Lemmatize user inputs to improve matching against vocabulary.	Limited to basic English morphology; does not handle domain-specific synonyms.
typing.Dict	Define and enforce expected structure of parsed JSON data.	Purely for type hints; no runtime validation.
json	Parse and serialize JSON data structures.	Assumes valid JSON; parsing errors are not caught.

Table 5 - Libraries and Limitations v1.3

3.2.2. Phase 2: Redesign The Conversational Agent

Up to this point, the prototype development has focused on generating LOIN documentation by guiding the user through fixed prompts, with limited success. This constrained setup prevented the AI from fully leveraging its reasoning over the standards and the ontology.

To overcome these constraints, Phase 2 shifts from a rigid, focused generator to a broader conversational agent. The prototype has been redesigned from the ground up to build a baseline RAG agent capable of handling natural queries across the full text of ISO 19650 and ISO 7817-1, and to layer complementary functionalities that refine the interaction.

The architecture has been reworked into clear modules, each responsible for a specific step or function. By replacing the former single-file script with a modular structure, the prototype remains better organized and becomes easier to adapt and extend in future versions.

The first stable version adopts a retrieval-first workflow: it queries the vector store for contextually relevant passages from the standards and then invokes the language model using this curated context. By decoupling retrieval and generation, the agent targets more precise, standards-aligned answers.

3.2.2.1. Stable Version 2.1:

This release (Appendix 1, v1.1.1) starts moving from fixed prompts to natural, retrieval-first dialogue over the indexed standards corpus. It keeps the focus entirely on local RAG and treats the agent as a lean baseline for answering free-form questions with passages drawn from ISO/EN sources.

At runtime, the app loads environment/config, binds the existing Chroma vector store (text-embedding-3-large), and configures a retriever to return the top passages for each user query. The flow is: user prompt, then similarity search across the indexed markdown of the standards, then assemble the retrieved excerpts as context, then call the chat model (GPT-4o-mini) and last, return an answer. Interaction is via a simple CLI loop; history is recorded but not yet fed back into prompts.

This local-only baseline improves flexibility versus earlier field-driven scripts but carries clear limits: no explicit ontology parsing, no structured field/LOIN validation, and knowledge integration based purely on passage similarity (no hierarchy/SHACL checks). These constraints make formal table extraction and semantic verification difficult, defining the gap later versions address. (See the pipeline diagram for Stable Version 2.1.)

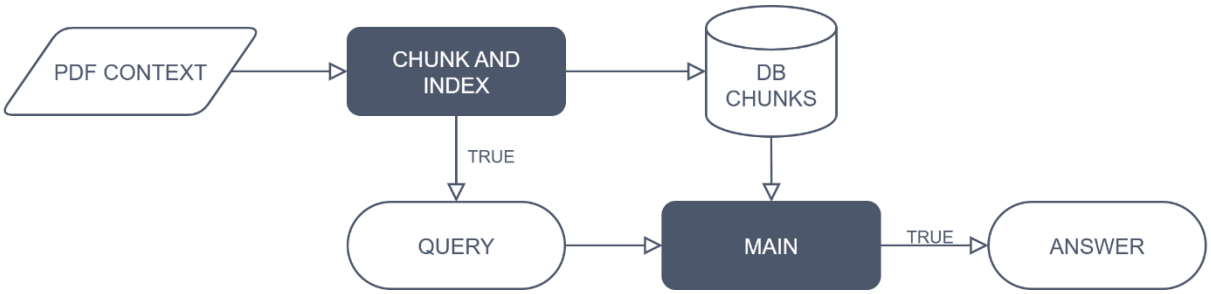


Figure 8 - Pipeline Diagram for Stable Version 2.1

The prototype's interaction is significantly more flexible than earlier function-based versions. Instead of guiding users through predefined fields, it lets them formulate questions naturally and uses embedded standard content to fetch context. This shift enhances adaptability and user autonomy.

However, the ontology is no longer explicitly parsed, and no structured field validation is present. Without field alignment, it becomes harder to extract formal LOIN tables or validate semantic correctness. Furthermore, the knowledge integration relies entirely on passage similarity, not ontology hierarchy or SHACL consistency. Real-time validation, multilingual support, and contextual disambiguation remain unimplemented.

Table 6 shows the Libraries and Modules used in this prototype aiming identify the limitations on their implementation.

Library / Module	**Role**	**Limitation**
os, sys, glob	File and system path handling, document listing.	No error checking for missing or malformed files.
uuid4	Unique ID generation for documents.	Not essential in this version; used for Chroma indexing.
dotenv.load_dotenv	Loads API keys and configuration variables from .env.	No fallback if environment variables are missing.
fitz (PyMuPDF)	PDF reading and text extraction.	Strips formatting; not robust to scanned or image-based documents.
langchain.schema.Document	Wraps chunks of text as LangChain Document objects.	Structure only; does not enrich documents semantically.
langchain.text_splitter.Markdown TextSplitter	Splits markdown into passages for embedding.	May split across semantically connected paragraphs.
langchain.vectorstores.Chroma	Stores and retrieves vector embeddings.	In-memory and local only; no distributed indexing.
langchain_openai.OpenAI Embeddings	Generates text embeddings using OpenAI models.	API-dependent; quality varies by model version.
langchain_openai.ChatOpenAI	Handles LLM queries with GPT-4o-mini.	Fixed model and temperature; no adaptive behavior.

Table 6 - Libraries and Limitations v2.1

3.2.2.2. Stable Version 2.2:

This version (Appendix 1, v.2.1.2) introduces a fully modular architecture, marking a key transition from earlier monolithic prototypes to a system where each core function is managed by an independent module. The main outcome of this version is to enable reliable RAG using only locally indexed PDF standards as the knowledge base, breaking down the process into clearly defined stages.

Each module in the pipeline is designed to handle a specific task. Starting from the transformation of raw PDF standards into a format suitable for downstream processing, and continuing through chunking, embedding, semantic retrieval, and context assembly. This modular structure not only clarifies the flow of data and control across the system but also makes it possible to adapt, extend, or replace individual components as requirements evolve. The interactions between modules are explicit, with each providing well-defined inputs and outputs to the next stage in the process.

At this point, development also begins on the first functionality layered on top of the RAG this functionality aims to work as a controlled web-retrieval path restricted to predefined domains. The addition is motivated by observations on the prototype's impossibility to give consistent answers to BIM related questions out of the scope of the documents.

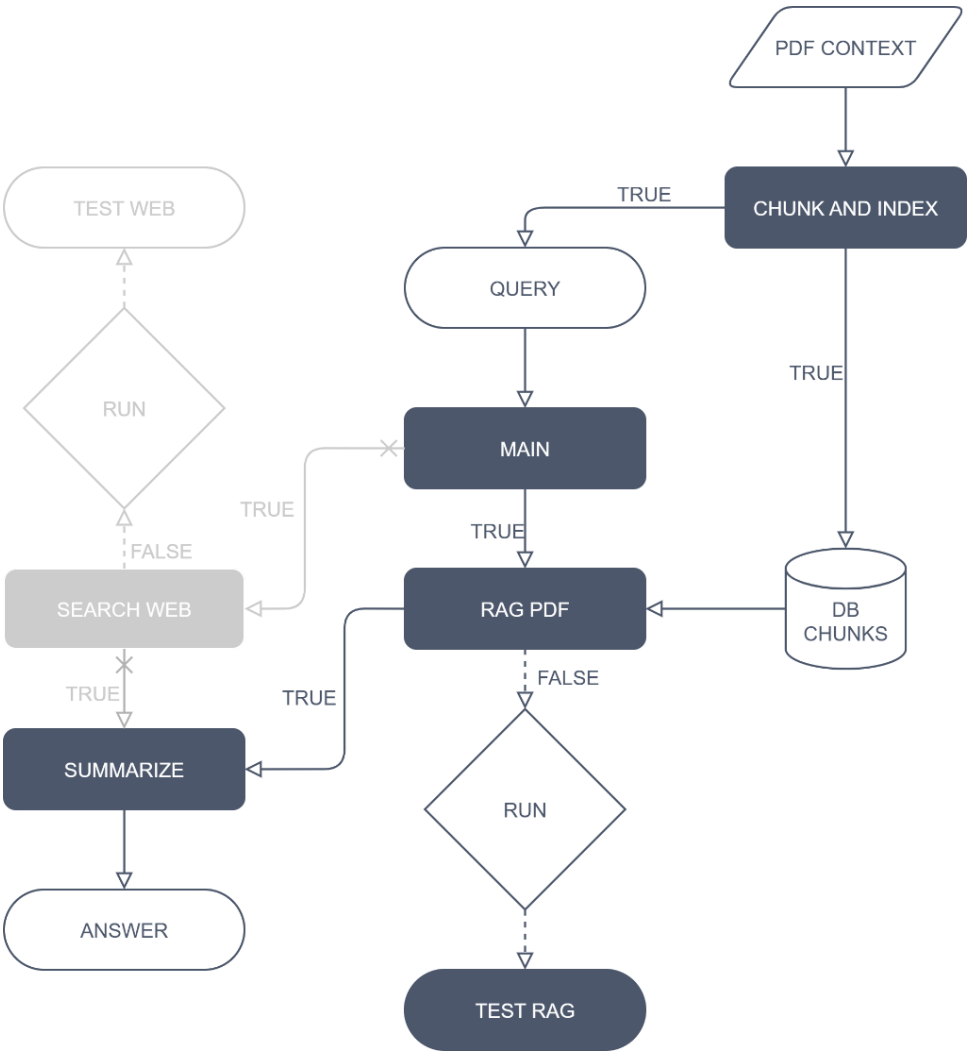


Figure 9 - Pipeline Diagram for Stable Version 2.2

The transformation of the architecture of the project sets the foundation for better flexibility and maintainability in future versions and enables the agent to process and respond to user queries with richer, document-grounded context than in previous releases. The overall flow and interactions between modules are illustrated by Figure 9.

The following subsections describe each module of the prototype and its interaction within the pipeline. For every module, the text states the role in the overall flow, the inputs/outputs and hand-offs, the principal libraries employed, and the current limitations. A standardized table accompanies each description with the columns Library/Module, Role, and Limitation:

a) Main Module (main.py)

This module serves as the central entry point for the application. It orchestrates the sequence of operations, handles user input, manages configuration, and calls the appropriate downstream modules for chunking, retrieval, prompt composition, and summarization. The main module is also responsible for handling program flow, launching CLI, and integrating optional or development modules. Limitations include the presence of legacy or placeholder functions and a degree of procedural logic that will benefit from further modularization in future versions.

Library / Module	**Role**	**Limitation**
sys, time	System operations, timing, and user interface control.	Only basic CLI, blocking waits, no GUI.
requests	Web requests (used for WebSearch, not main pipeline).	External dependency; unreliable for scraping.
bs4 (BeautifulSoup)	HTML parsing for WebSearch (inactive in stable pipeline).	Not needed in main workflow in this version.
modules.rag_pdf	Calls retrieval and answer generation pipeline.	Relies on downstream module implementation.
modules.search_web	Placeholder for online retrieval (inactive in this version).	Not integrated in stable workflow.
modules.summarize	Calls summarization function.	Optional, variable results.
modules.prompt_composer	Assembles prompt for language model.	Template-based, limited adaptability.
config	Loads configuration and environment variables.	Static; requires correct environment setup.

Table 7 - Libraries and Limitations Main v2.2

b) Chunking and Indexing Module

This module is responsible for preparing the knowledge base from raw standard documents. It begins by reading and extracting text from PDF files, then converts this content to markdown format. The module splits the text into manageable passages, generates vector embeddings for each chunk, and indexes them in a Chroma vector store. All downstream retrieval depends on the quality and organization of this initial process. Key limitations include possible loss of document structure, challenges with scanned or image-based PDFs, and a reliance on external APIs for embedding.

Library / Module	Role	Limitation
os, glob	File and directory management.	Limited error handling, no advanced file checks.
fitz (PyMuPDF)	Extracts text from PDFs.	Strips formatting, fails on image-only/scanned PDFs.
uuid4	Generates unique IDs for document storage.	Not critical to core logic.
dotenv	Loads environment variables.	Fails if env missing/incomplete.
langchain.schema.Document	Wraps text chunks as structured documents.	Structure only, no semantic metadata.
langchain.text_splitter.MarkdownTextSplitter	Splits markdown files into manageable passages.	May break logical units; fixed chunking.
langchain_openai.OpenAIEmbeddings	Embeds text passages as vectors.	API dependence, external cost/latency.
langchain_chroma.Chroma	Stores and retrieves vectors.	In-memory/local only, no distributed scaling.
chromadb.config.Settings	Configuration for Chroma vector store.	Static configuration, must match environment.

Table 8 - Libraries and Limitations Indexing v2.2

c) Retrieval Module

This module manages semantic search across the indexed corpus. When a user query is received, it performs a similarity search within the local vector store and retrieves the most relevant text passages. The module is limited by its dependence as it's running on local content only; online or external sources are not fully incorporated in this version.

Library / Module	Role	Limitation
langchain.chains.retrieval_qa.base.RetrievalQA	Manages end-to-end retrieval-augmented QA chain.	Single-pass retrieval.
langchain.prompts.PromptTemplate	Formats prompt for the language model.	Template logic, limited flexibility.
langchain_core.documents.Document	Wraps retrieved passages as structured documents.	Only basic metadata, no ontology awareness.
langchain_chroma.Chroma	Searches indexed document vectors.	Only searches local documents.
langchain_openai.OpenAIEmbeddings	For semantic retrieval, embeddings.	Dependent on OpenAI, external cost.
chromadb.config.Settings	Configuration for Chroma vector store.	Needs manual/environment setup.
config	Provides runtime configuration.	Static settings, may require updates.

Table 9 - Libraries and Limitations RAG v2.2

d) Prompt Composer

After retrieval, this module assembles the selected passages into a structured context prompt for the language model.

Library / Module	Role	Limitation
(custom code only)	Assembles structured prompt for LLM.	Fixed templates, lacks adaptability.

Table 10 - Libraries and Limitations Composer v2.2

e) Summarization

This optional module further condenses retrieved or generated content, using a language model to provide concise outputs. Its integration is not mandatory in the main workflow and may produce variable results.

Library / Module	Role	Limitation
config	Loads language model for summarization.	Summarization quality may vary.

Table 11 - Libraries and Limitations Summarize v2.2

f) WebSearch Module (under development)

Although present in the codebase, this module is under development in this version. It was designed to supplement responses with online information but is not able to retrieve information into the stable workflow.

Library / Module	Role	Limitation
time	Used for timing, retries.	Basic, not robust.
requests	Fetches web content for search.	Not reliable, no error handling.
bs4 (BeautifulSoup)	Parses HTML for extracting search results.	Not needed in stable pipeline in this version.
config	Loads search configuration.	Not integrated; inactive module.

Table 12 - Libraries and Limitations WebSearch v2.2.

3.2.2.3. Stable Version 2.3:

This version (Appendix 1, v2.2.2) marks a major evolution of the agent, expanding its modular architecture to integrate both local document retrieval and functional WebSearch capabilities. The core design principle remains unchanged: each function within the system is encapsulated in an independent module, ensuring clear separation of concerns and explicit data flow between processing stages. As illustrated on Figure 10, the pipeline now supports not only RAG from locally indexed PDF standards but also dynamic supplementation with online context when the internal knowledge base is insufficient.

Each module in the workflow addresses a specific aspect of the information retrieval process. The pipeline begins with the transformation of raw PDF standards into structured markdown, followed by

chunking and embedding these passages for efficient semantic search. When a user query cannot be fully resolved using local content, the system seamlessly invokes the WebSearch and DeepSearch modules, issuing online queries, refining results through iterative rephrasing, and integrating new context into the response. As in earlier versions, modules for prompt composition and summarization ensure that all retrieved information, local or online, is presented in a coherent and user-focused format.

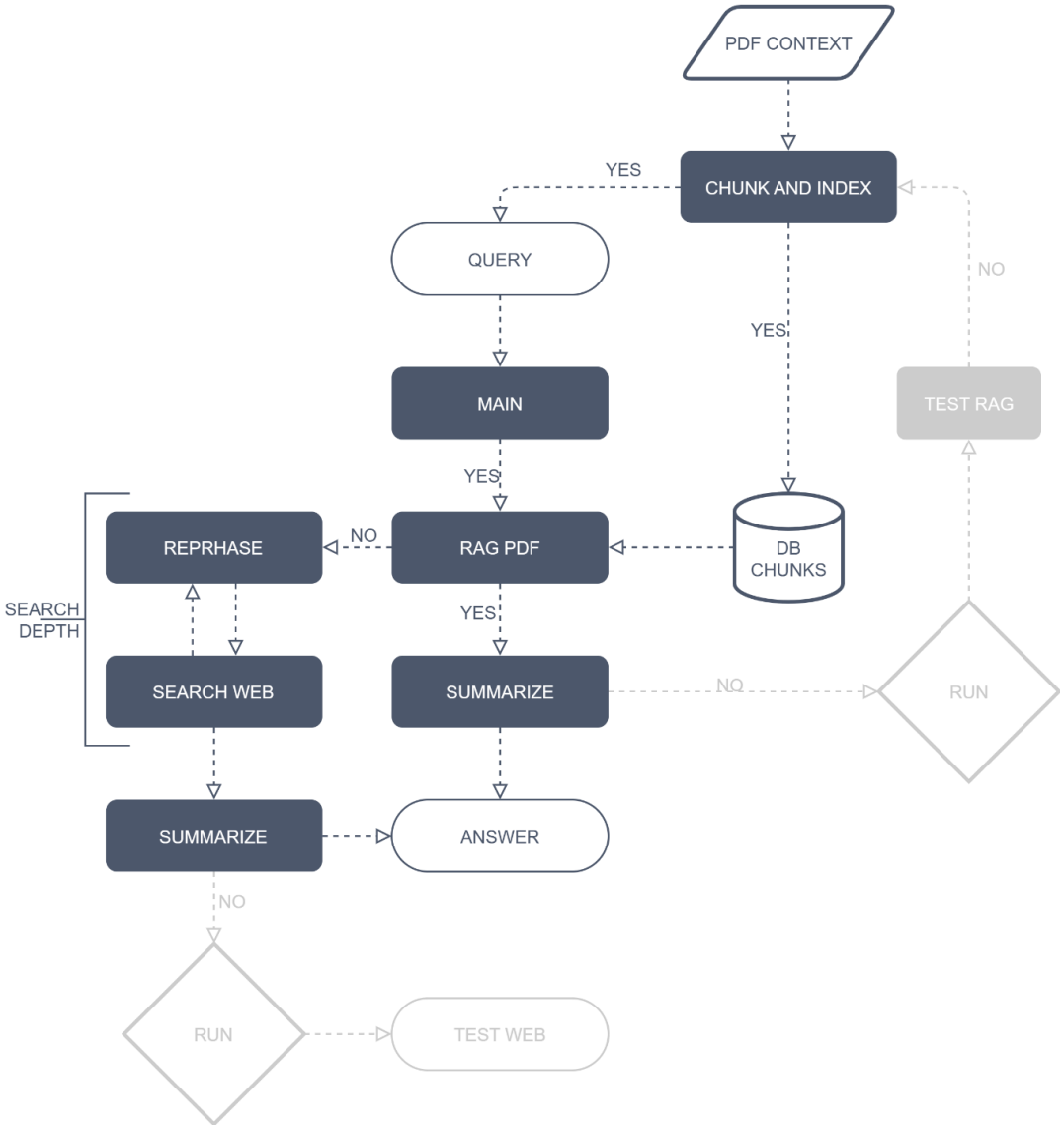


Figure 10 - Pipeline Diagram for Stable Version 2.3

This flexible, hybrid architecture positions the agent to deliver context-rich answers even as new standards emerge or as knowledge in the construction domain evolves. The explicit modularity not only enhances maintainability and extensibility but also supports ongoing experimentation with advanced retrieval and reasoning techniques.

The following subsections describe each module of the prototype and its interaction within the pipeline. For every module, the text states the role in the overall flow, the inputs/outputs and hand-offs, the principal libraries employed, and the current limitations. A standardized table accompanies each description with the columns Library/Module, Role, and Limitation:

a) Main:

This script serves as the primary entry point for the prototype application. It manages the overall workflow, including parsing user input, loading environment configuration, initializing logging, and orchestrating the sequence of module calls (chunking, retrieval, prompt composition, WebSearch, and summarization). The main module determines, based on context sufficiency, whether to respond using only local standards or to invoke hybrid retrieval with online sources.

Development Note: Compared to earlier versions, main.py now integrates hybrid retrieval logic and enhanced session management, supporting more robust and modular execution.

Library/Module	**Role**	**Limitation**
app.rag_pdf.fetch_pdf_knowledge	Retrieves local PDF context (standards corpus) for the user query; first-pass answer source.	Quality depends on prior chunking/indexing; may return long, unranked context; no fallback if index is stale.
app.search_web.run_web_search	Performs site-specific WebSearch over TRUSTED_DOMAINS; used when local PDF context is insufficient.	Network/latency sensitive; fallback=False can yield empty results; domain list must be curated.
app.summarize.summarize_answer	Composes the final answer given the user query, optional pdf_ctx, and web hits; also used to summarize per-domain results.	No explicit token budgeting; output quality tied to LLM and input context cleanliness.
app.logger.log_interaction	Persists interaction data (query, contexts, response, history, search depth) for traceability.	File/log sink not shown here; potential PII if not sanitized; limited analytics without postprocessing.
config.llm	LLM client used to (a) rewrite queries for BIM-specific search and (b) decompose into subquestions when no hits.	External API dependence (keys/cost/limits); .invoke() responses must be sanitized; model choice affects style/quality.
config.TRUSTED_DOMAINS	Whitelist of BIM-relevant domains; drives site-prefixed queries in the web branch.	If incomplete/outdated, relevant content is missed; maintenance burden.
config.SEARCH_DEPTH	Recorded with each interaction to document depth/effort of search.	Static per run; not adapted to query difficulty.

Table 13 - Libraries and Limitations Main v2.3

b) Chunk and Index:

This module prepares the knowledge base by ingesting standard documents in PDF format, extracting their content, converting the text to markdown, splitting the text into passages, generating vector embeddings, and indexing these embeddings in a Chroma vector store. This process ensures all downstream retrieval queries have access to semantically chunked and indexed standard content.

Development Note: While the core workflow is consistent with prior versions, recent improvements include expanded document support, a more structured database directory, and enhanced modularity.

Library / Module	**Role**	**Limitation**
os, sys, glob	File ops, path handling, and file discovery for PDFs/MDs.	Minimal error handling; fragile on inconsistent dirs.
fitz (PyMuPDF)	PDF text extraction used to build Markdown with pseudo-headings.	Loses layout; weak on scanned/image PDFs.
uuid4	Generates unique IDs when needed.	Bookkeeping only; no effect on retrieval quality.
pathlib.Path	Cross-platform path resolution; sets PROJECT_ROOT.	None significant; misuse can break sys.path insertion.
config (DATA_PATH, MD_OUTPUT_PATH, CHROMA_PATH, COLLECTION)	Centralizes input/output dirs and Chroma collection name.	Misconfiguration halts pipeline; no autovalidation.
langchain.schema.Document	Wraps chunks with metadata (source, chunk) before indexing.	Minimal schema; no semantic enrichment.
langchain.text_splitter.MarkdownTextSplitter	Splits Markdown into chunks (size 1000, overlap 200).	Fixed rules can split concepts; needs tuning per corpus.
langchain_openai.embeddings.OpenAIEmbeddings	Creates vector embeddings (text-embedding-3-large).	External API cost/latency; model/version dependency.
langchain_chroma.Chroma	Vector store client for add/search over embeddings.	Local store; no distributed scaling by default.
chromadb.config.Settings	Configures Chroma (persist dir, telemetry off).	Static settings; environment-sensitive paths.

Table 14 - Libraries and Limitations Chunk and Index v2.3

c) Conversational LOIN:

Implements conversational logic for the agent, enabling multi-turn dialogue with users. The module manages session state and ensures that each user query is processed within an interactive, context-aware conversation, coordinating the retrieval and generation steps.

Development Note: This conversational interface is a new addition in this release, reflecting a shift from single-turn command-line interaction to a more dynamic, user-friendly experience.

Library / Module	Role	Limitation
openai	Interface with OpenAI API for conversational completions.	Dependent on external API, latency, and cost.
dotenv.load_dotenv	Loads API keys and environment variables from .env.	Requires manual .env setup; potential security risk if mismanaged.
os	Accesses environment variables (API keys, configuration).	No validation of missing/invalid keys.
logging	Handles logs for debugging and tracing dialogue flow.	Basic setup; no advanced log management.
json	Stores/loads conversation state in structured format.	No schema validation, may break if malformed.
time	Adds timing/delays in API calls.	Only basic control, not async optimized.
sys	Handles exit/error termination.	Minimal error granularity.
collections.deque	Manages conversational history buffer.	Limited to in-memory; no persistence.
dataclasses.dataclass	Defines structured containers for LOIN conversation fields.	Only runtime structure; no persistence or validation layer.

Table 15 - Libraries and Limitations Conversational LOIN v2.3

d) Prompt Composer:

Responsible for assembling retrieved passages (from both local and online sources) into a structured prompt for the language model. This ensures that the agent's outputs are coherent, aligned to the standards, and rich in context.

Development Note*:*
Enhanced in this version to support hybrid retrieval, integrating WebSearch results seamlessly with local document context.

Library / Module	Role	Limitation
(custom logic only)	Session handling and turn management (tracking responses, checking completeness, summarizing context).	No external state store; minimal recovery, limited persistence beyond runtime.

Table 16 - Libraries and Limitations Prompt Composer v2.3

e) RAG:

Executes RAG using the indexed PDF corpus. It searches the Chroma vector store for the most relevant chunks, prepares these passages for the prompt composer, and signals the need for online augmentation if local context is insufficient.

Development Note*:*
Integration with the new WebSearch and DeepSearch modules is now supported, improving the overall adaptability of the retrieval pipeline.

Library / Module	**Role**	**Limitation**
os, sys, pathlib.Path	Path management and sys.path injection to import configuration from project root.	Fragile if folder structure changes; sys.path mutation can mask import issues.
warnings	Suppresses noisy LangChain retriever warnings.	Can hide useful deprecation/runtime hints.
typing.List, typing.Set	Type hints for documents and de-duplication.	None at runtime (typing-only).
config.llm, config.CHROMA_PATH, config.COLLECTION, config.NUM_RESULTS	Externalized configuration for LLM client and vector store settings, and top-k retrieval.	Misconfiguration halts retrieval; static NUM_RESULTS not adaptive to query difficulty.
langchain.schema.Document	Document type used by retriever and QA chain.	Minimal metadata; no schema/ontology enrichment.
langchain_openai.OpenAIEmbeddings	Generates embeddings (text-embedding-3-large) for Chroma search.	External API cost/latency; vendor/model dependency; requires network.
langchain_chroma.Chroma	Local vector store; used with .as_retriever(search_type="mmr").	Local persistence only; limited horizontal scaling; performance depends on disk/host.
chromadb.config.Settings	Configures Chroma client (persist dir, telemetry off).	Static, environment-sensitive paths; no runtime validation.
langchain_openai.ChatOpenAI	LLM client for optional QA chain (gpt-4, temperature 0).	API costs/limits; hidden prompt logic inside chain; network required.
langchain.chains.question_answering.load_qa_chain	Optional *stuff* chain over raw Document objects (get_pdf_answer).	Blackbox prompts; token-heavy on long contexts; not used in main retrieval path.

Table 17 - Libraries and Limitations RAG v2.3

f) WebSearch:

Implements the WebSearch and DeepSearch logic, retrieving supplementary information from online sources. Capable of performing iterative (multi-depth) search and rephrasing queries to enhance retrieval diversity and depth.

Development Note*:*

This module transitions from an inactive or placeholder role in previous versions to an operational component, marking a significant extension of the agent's retrieval scope.

Library / Module	**Role**	**Limitation**
ddgs.DDGS	DuckDuckGo search client used to fetch text results for a query.	Third-party service; result fields vary (href/url/link); may throttle or change API behavior.
urllib.parse.urlparse	Extracts domain from result URLs to enforce trusted domain filtering.	Simple string-based check; can be bypassed by redirects or CDN domains.
requests	Downloads HTML for pages selected from search results.	Network latency/failures; timeouts; no JS rendering (static HTML only).
bs4.BeautifulSoup	Parses HTML and extracts text from <p>, , <h2>, <h3>.	Limited to static DOM; brittle on malformed HTML; ignores dynamic content.
config.USER_AGENT	Custom UserAgent header for HTTP requests.	Must be maintained to avoid blocking; misconfiguration can cause 403s.
config.TOP_K_URLS	Upper bound of search results requested from DDG.	Static cap; may miss relevant results if set too low.
config.TRUSTED_DOMAINS	Whitelist for site filtering; only those domains are returned unless fallback.	Requires ongoing curation; strict lists reduce recall; subdomain handling is simplistic.

Table 18 - Libraries and Limitations Search Web v2.3

g) Summarize:

Offers optional summarization of generated responses, using a language model to condense content and improve readability for end users.

Development Note*:*

Refined integration allows the module to summarize both local and online content.

Library / Module	**Role**	**Limitation**
openai	Calls LLM API to generate summaries.	External dependency; subject to API cost and latency.
dotenv.load_dotenv	Loads environment variables (e.g., API keys).	Fails silently if .env misconfigured.
os	Access to system environment variables.	Limited to environment context.

json	Parse and format structured outputs.	Limited handling for malformed JSON.
logging	Tracks summarization process and errors.	Only local logs; no advanced monitoring.

Table 19 - Libraries and Limitations Summarize v2.3

h) Logger:

Provides systematic logging of session activities, errors, and significant events throughout the application's execution. Logs are stored for troubleshooting, traceability, and process audit purposes.

Development Note*:*
Standardizes and formalizes logging compared to ad-hoc or absent logging in previous versions.

Library / Module	**Role**	**Limitation**
os	Manages directories and file paths for logs.	Platform-dependent path handling; minimal error reporting.
datetime	Timestamps logs for daily session files.	Limited to system time; no time zone normalization.
(custom logic)	Writes interaction history, PDF context, web summaries, and responses into daily session files.	No structured log format (e.g., JSON); limited querying or filtering capabilities.

Table 20 - Libraries and Limitations Logger v2.3

3.2.2.4. Stable Version 2.4:

The current version of the development roadmap (v2.2.4) of the prototype, represents a consolidation stage in the development of prototype. Positioned after the foundational modularization introduced in Stable Version 2.3, this release focuses on refining the RAG architecture and improving the reliability of both local and external search components. The version maintains the modular structure adopted in the previous iteration while addressing limitations in integration and execution flow, preparing the groundwork for subsequent enhancements in reasoning and query handling.

Compared to the previous version, this release introduces targeted adjustments aimed at improving context retrieval accuracy and overall pipeline stability. The chunking and indexing process benefits from optimizations in text handling, reducing fragmentation, and improving semantic consistency of stored passages. The retrieval logic is streamlined to better prioritize relevant results from the local vector store, and the interaction between retrieval and prompt composition has been refined to ensure more coherent and richer in context model inputs. Additionally, the external search functions have been stabilized, with preliminary adjustments to better control when and how web-based retrieval is triggered.

Several capabilities introduced in the previous stable version have been refined to improve both performance and reliability. The RAG pipeline now demonstrates more predictable behavior under varied query types, with fewer irrelevant results and smoother handoff between modules. Improvements

in text preprocessing during the chunking stage contribute to higher retrieval precision, while adjustments in prompt composition reduce redundancy and improve alignment with the intended table output format. Furthermore, module interconnections have been clarified, resulting in a cleaner execution flow and reduced likelihood of errors during multi-step queries.

This version reintroduces and refines functionalities that extend the agent's versatility. The WebSearch capability, previously limited in scope, is now triggered alongside local retrieval, prioritizing content from the indexed standards while supplementing gaps with external sources. As shown in Figure 11, the pipeline has been restructured to coordinate module calls more efficiently, reducing unnecessary processing and enabling smoother integration between retrieval, prompt composition, and optional summarization. Minor updates to the chunking process improve the consistency of extracted text, while prompt formatting adjustments aim to minimize redundant information and maintain alignment with the intended output structure. Initial elements of a search depth mechanism have also been incorporated, laying the groundwork for future iterations capable of progressively refining queries based on prior results.

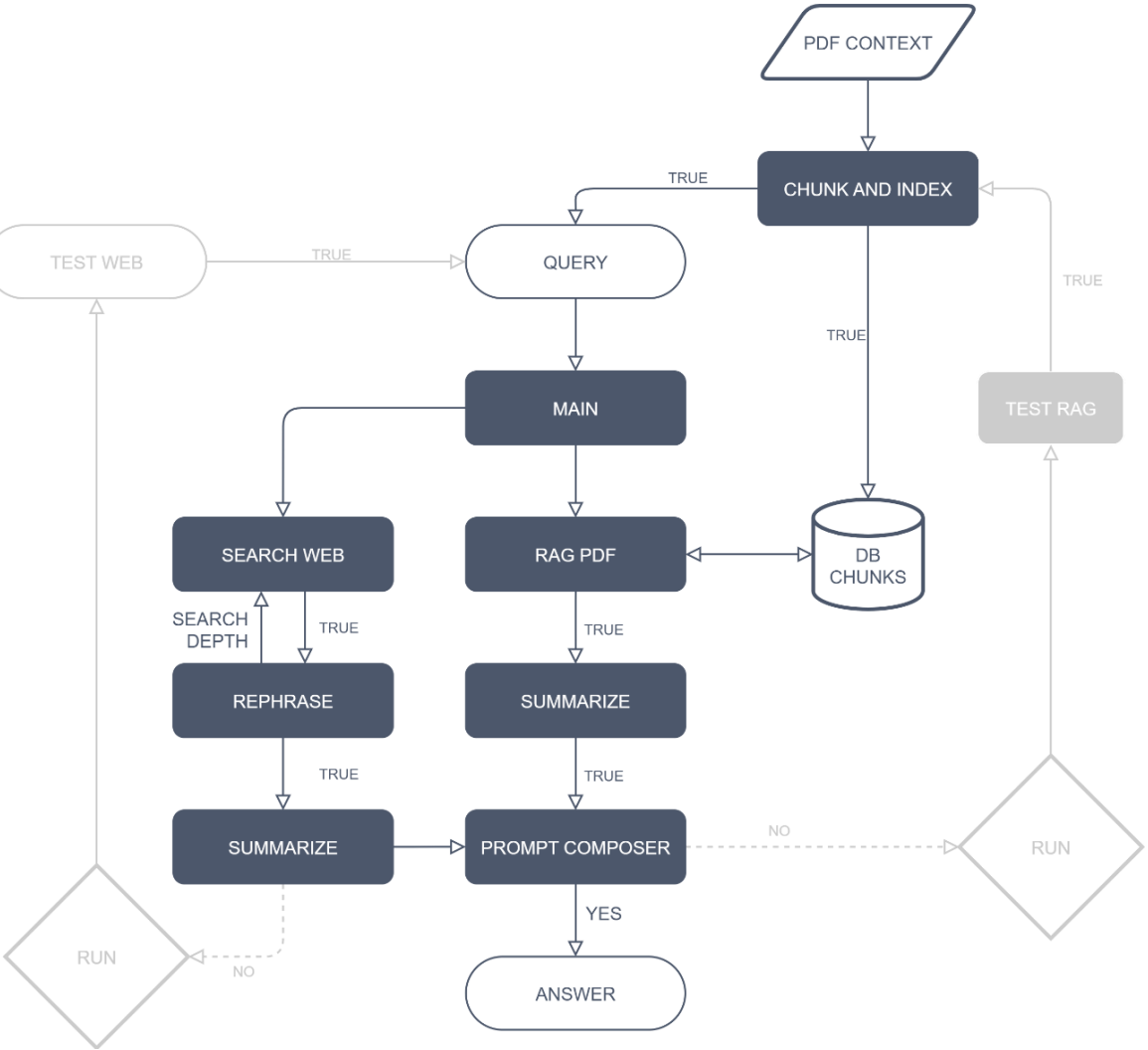


Figure 11 - Pipeline Diagram for Stable Version 2.4

Although this release delivers enhancements to the agent's modular architecture, several constraints remain evident. Error handling mechanisms are still basic, offering limited resilience in cases of missing files, API failures, or connectivity interruptions. While modularization has improved maintainability and scalability, further optimization of inter-module communication and computational efficiency will be necessary to accommodate more complex workflows in future iterations.

The following subsections describe each module of the prototype and its interaction within the pipeline. For every module, the text states the role in the overall flow, the inputs/outputs and hand-offs, the principal libraries employed, and the current limitations. A standardized table accompanies each description with the columns Library/Module, Role, and Limitation:

i) Main

This script serves as the primary entry point for the prototype application. It manages the overall workflow, including parsing user input, loading environment configuration, initializing logging, and orchestrating the sequence of module calls (chunking, retrieval, prompt composition, WebSearch, and summarization). The main module determines, based on context sufficiency, whether to respond using only local standards or to invoke hybrid retrieval with online sources.

Development Note: Compared to earlier versions, main.py now integrates hybrid retrieval logic and enhanced session management, supporting more robust and modular execution.

Library / Module	**Role**	**Limitation**
app.rag_pdf	Local retrieval over indexed ISO PDFs.	Dependent on embedding quality and DB freshness.
app.search_web	Executes online search workflow,	Network dependent; variable content quality.
app.logger	Records interactions and events.	File-based; limited analytics.
app.summarize	Optional answer summarization.	Output quality varies with LLM.
config.llm, SEARCH_DEPTH, TRUSTED_DOMAINS	LLM client and control params.	Static configuration; requires correct .env.

Table 21 - Libraries and Limitations Main v2.4

j) Chunk and Index

This module prepares the knowledge base by ingesting standard documents in PDF format, extracting their content, converting the text to markdown, splitting the text into passages, generating vector embeddings, and indexing these embeddings in a Chroma vector store. This process ensures all downstream retrieval queries have access to semantically chunked and indexed standard content.

Development Note: While the core workflow is consistent with prior versions, recent improvements include expanded document support, a more structured database directory, and enhanced modularity.

Library / Module	Role	Limitation
fitz (PyMuPDF)	PDF text extraction.	Loses layout; weak on scanned PDFs.
langchain.text_splitter. MarkdownTextSplitter	Passage chunking.	Fixed rules may split concepts.
langchain_openai.OpenAI Embeddings	Vector embeddings.	External API cost and latency.
langchain_chroma.Chroma	Vector index and retrieval.	Local store; no distributed scaling.
langchain.schema.Document	Wraps chunks with minimal metadata.	No semantic enrichment.
chromadb.config.Settings	Chroma configuration.	Static settings; env sensitive.
os, glob, uuid4, pathlib.Path, config	File discovery, IDs, paths.	Limited error handling.

Table 22 – Libraries and Limitations Chunk and Index v2.4

k) Conversational LOIN

Implements conversational logic for the agent, enabling multiturn dialogue with users. The module manages session state and ensures that each user query is processed within an interactive, context-aware conversation, coordinating the retrieval and generation steps.

Development Note: This conversational interface is a new addition in this release, reflecting a shift from single-turn command-line interaction to a more dynamic, user-focused experience.

Library / Module	Role	Limitation
(custom logic only)	Session handling and turn management.	No external state store; minimal recovery.

Table 23 - Libraries and Limitations Conversational Loin v2.4

l) Prompt Composer

Responsible for assembling retrieved passages (from both local and online sources) into a structured prompt for the language model. This ensures that the agent's outputs are coherent, aligned to the standards, and rich in context.

Development Note: Enhanced in this version to support hybrid retrieval, integrating WebSearch results seamlessly with local document context.

Library / Module	Role	Limitation
(custom logic only)	Formats context and instructions for LLM.	Template-based; limited adaptability.

Table 24 - Libraries and Limitations Prompt Composer v2.4

m) RAG

Executes RAG using the indexed PDF corpus. It searches the Chroma vector store for the most relevant chunks, prepares these passages for the prompt composer, and signals the need for online augmentation if local context is insufficient.

Development Note: Integration with the WebSearch and DeepSearch routines is supported, improving adaptability of the retrieval pipeline.

Library / Module	**Role**	**Limitation**
langchain_chroma.Chroma	Retrieves top passages from local DB.	Local only; index consistency required.
langchain_openai.OpenAI Embeddings	Embedding space for search.	API dependency; version-sensitivity.
langchain.schema.Document	Document wrappers for retrieved chunks.	Minimal metadata.
langchain.chains.question_ answering.load_qa_chain	LLM QA chain utility.	Adds overhead; limited control.
config (llm, paths, constants)	LLM client and paths.	Requires correct env/configuration.
os, sys, pathlib.Path, warnings, typing	Runtime setup and hygiene.	—

Table 25 - Libraries and Limitations RAG v2.4

n) WebSearch

Implements the WebSearch and DeepSearch logic, retrieving supplementary information from online sources. Capable of performing iterative search and rephrasing queries to enhance retrieval diversity and depth.

Development Note: Transitions from placeholder to operational use; prioritization now favors local RAG while always running WebSearch in parallel.

Library / Module	**Role**	**Limitation**
requests	Fetches web pages.	Network- and site policy-dependent.
bs4.BeautifulSoup	HTML parsing and text extraction.	Structure-dependent; may include noise.
ddgs.DDGS	DuckDuckGo search API client.	Third-party dependency; rate limits.
urllib.parse.urlparse	Domain filtering.	Simple heuristics; not foolproof.
config (USER_AGENT, TOP_K_URLS, TRUSTED_DOMAINS)	Search configuration and filters.	Static lists; manual maintenance.

Table 26 - Libraries and Limitations Search WEB v2.4

o) Summarize

Offers optional summarization of generated responses, using a language model to condense content and improve readability for end users.

Development Note: Refined integration allows the module to summarize both local and online content.

Library / Module	Role	Limitation
config.llm	LLM client used for summarization.	Quality varies with prompt and model.
app.prompt_composer. compose_final_prompt	Builds summary prompt.	Template constraints.

Table 27 - Libraries and Limitations Summarize v2.4

p) Logger

Provides systematic logging of session activities, errors, and significant events throughout the application's execution. Logs are stored for troubleshooting, traceability, and process audit purposes.

Development Note: Standardizes and formalizes logging compared to ad hoc or absent logging in previous versions.

Library / Module	Role	Limitation
os, datetime.datetime	File based logs with timestamps.	Local files only; no analytics backend.

Table 28 - Libraries and Limitations Logger v2.4

3.2.2.5. Stable Version 2.5: Current State of the Prototype.

This release (v2.2.4.3) represents the most recent and stable version state in the development of the prototype. It builds directly upon Stable Version 2.4, aiming to consolidate the functionalities introduced previously while addressing structural and operational refinements. This design enables a systematic evaluation of how the additional modules, such as WebSearch, summarization, or conversational management, impact the baseline RAG pipeline built exclusively on the indexed ISO standards. As such, stable version 2.5 represents the current reference point in the project timeline, establishing the foundation for further experimentation, validation, and extension of the prototype.

In comparison with Stable Version 2.4, Stable Version 2.5 is characterized by significant architectural and functional changes that extend the prototype beyond its initial baseline. While the previous version consolidated a RAG pipeline centered on the local content, the new release broadens this foundation through the introduction of specialized agents and structured knowledge components.

Among the most relevant modifications is the incorporation of a conversational orchestration layer, designed to preserve context and manage multi-turn exchanges with greater semantic stability. This addition is complemented by the development on the DeepSearch mechanisms, operating both locally and through online iterative exploration, aiming to enrich the information retrieval process by connecting the standard-based reasoning with external, high-quality sources. Equally important is the

formalization of LOIN management through schema definitions and ISO 7817-aligned templates, with the goal of enabling a systematic capture and validation of information needs in line with the principles defined in ISO 7817-1 and ISO 19650-1.

Furthermore, ontology modules have been integrated for the first time in this second phase, reintroducing the semantic representation of LOIN concepts that facilitates machine-interpretable knowledge alignment in accordance with ISO 23387. Together, these enhancements represent a decisive step toward coupling procedural information management with ontological reasoning, thereby aiming to reinforce both the methodological consistency and the extensibility of the prototype. Alongside the introduction of new modules, Stable Version 2.5 also delivers substantial refinements to previously established functionalities, with particular emphasis on optimization, modularization, and operational stability. The summarization mechanisms, which were already present in the earlier release, have been restructured into specialized routines such as simplified summaries and targeted LOIN summarization, thereby improving their consistency and adaptability across heterogeneous inputs. Figure 12 illustrate the changes on the structure and workflow of stable version 2.5.

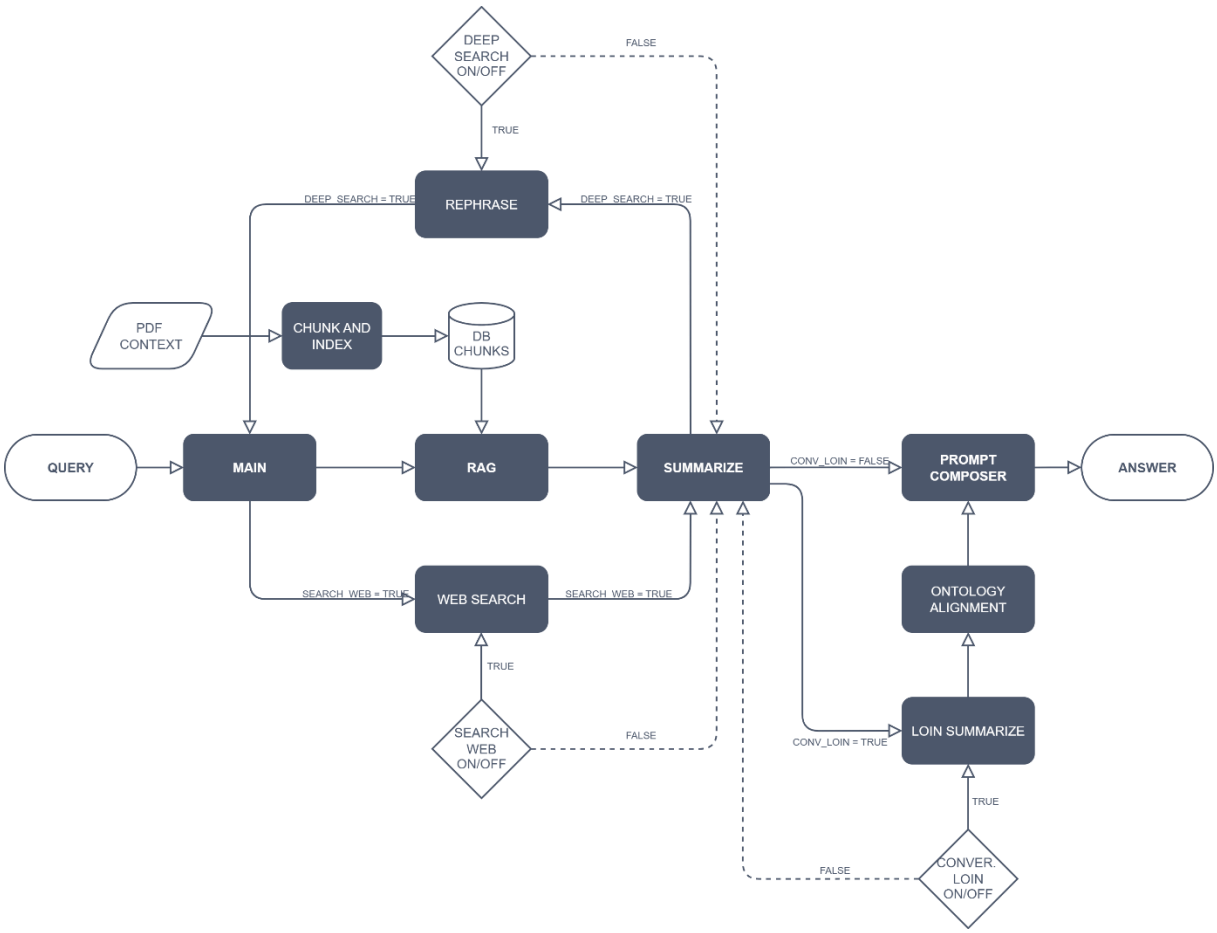


Figure 12 - Pipeline Diagram for Stable Version 2.5

In parallel, the configuration framework has been expanded to incorporate granular feature toggles, enabling controlled activation of advanced functions such as DeepSearch or conversational LOIN management. This approach was not only conceived to preserve the option to execute reduced and standard-focused baselines but also to systematically evaluate how different strategies affect the overall performance and capabilities of the agent. These refinements also extend to the internal orchestration of the pipeline, where execution flows have been modularized to reduce redundancy, enhance maintainability, and ensure more predictable performance under varying usage conditions.

The following subsections describe each module of the prototype and its interaction within the pipeline. For every module, the text states the role in the overall flow, the inputs/outputs and hand-offs, the principal libraries employed, and the current limitations. A standardized table accompanies each description with the columns Library/Module, Role, and Limitation:

a) Main

This script serves as the central entry point for the 2.2.4.3 version of the prototype. It initializes environment flags, loads configuration, and orchestrates the full multi-stage pipeline: local RAG retrieval, optional DeepSearch (local or online), trusted web retrieval, summarization, and conversational LOIN workflows if enabled. The module incorporates flexible configuration switches to activate or deactivate submodules (e.g., Conversational LOIN, Ontology layer, DeepSearch. It also integrates logging utilities for capturing interactions and ensures graceful fallback when optional components (like ontology or conversation tools) are unavailable.

Development Note: Compared to the earlier 2.2.2 and 2.2.4 versions, this release introduces a more advanced multi-stage pipeline, explicit conversational branching, ontology cache integration, and improved handling of on/off flags. These refinements make the execution flow more robust and configurable while maintaining resilience when optional modules are missing.

Library / Module	**Role**	**Limitation**
typing (List, Tuple, Optional)	Type hinting and function signatures.	Only static type support, no runtime enforcement.
app.rag_pdf.fetch_pdf_knowledge	Local retrieval from PDF vector index.	Dependent on chunk/index pipeline; context may be limited.
app.search_web.run_web_search	WebSearch for external content.	Coverage depends on search engine and trusted domains.
app.logger.log_interaction	Logging user queries, context, and responses.	No external monitoring integration.
app.summarize.summarize_simple, app.summarize.summarize_loin	Summarization of responses (generic vs LOIN-focused).	Quality depends on LLM; no deterministic outputs.
config (flags + llm)	Global configuration and LLM binding.	Static values at startup; requires manual tuning.

app.deepsearch.local_deepsearch (optional)	Local iterative query refinement.	May fail silently; not always available.
app.deepsearch_online.web_depresearch (optional)	Online DeepSearch with iterative re-querying.	External calls; slower; depends on LLM reliability.
app.convo_agent (optional: multiple functions)	Conversational LOIN session handling, inference, ontology integration.	Disabled if modules missing; fragile if ontology unavailable.

Table 29 – Libraries and Limitations Main v2.5

b) Chunk and Index

This module processes standard documents (PDFs) into semantically indexed chunks. It converts PDFs to Markdown with structural hints (page markers, headings), applies token-aware chunking, tags content with LOIN and ontology terms, and indexes the resulting chunks into a Chroma vector store with OpenAI embeddings. It also maintains a manifest to avoid re-embedding unchanged documents.

Development Note: Compared to earlier versions, this release introduces ontology-aware tagging, token-aware chunking with overlap, and manifest-based incremental indexing. It integrates rdflib-based ontology ingestion when enabled.

Library / Module	**Role**	**Limitation**
fitz (PyMuPDF)	Extracts structured text, spans, and font sizes from PDFs for Markdown conversion.	Loses complex layout, weak on scanned/image-based PDFs.
langchain.schema.Document	Wraps text chunks with metadata for retrieval.	Minimal schema, no built-in validation.
langchain_chroma.Chroma	Vector store backend for embeddings.	Local-only; no distributed scaling.
langchain_openai.OpenAIEmbeddings	Embedding generation for chunks.	External API, cost, and latency dependent.
chromadb.config.Settings	Configures Chroma persistence.	Static configuration, environment-sensitive.
tiktoken (optional)	Token-aware chunking for size-controlled splits.	Dependency may fail; falls back to naive character count.
rdflib.Graph (optional, if ontology enabled)	Loads and parses ontology terms from TTL files.	Parsing overhead; requires consistent ontology schema.
os, sys, glob, json, hashlib, pathlib.Path, uuid4, re, csv	File discovery, hashing, manifesting, metadata tagging.	Limited error handling; lightweight CSV parsing only.

Table 30 - Libraries and Limitations Chunk and Index v2.5

c) RAG

This module retrieves semantically relevant passages from the locally indexed standards corpus and prepares them as grounded context for downstream generation. It initializes a persistent Chroma vector store with OpenAI embeddings, exposes a *Maximal Marginal Relevance* (MMR) retriever to balance relevance and diversity, and concatenates unique chunks with lightweight provenance (source filename).

An optional QA chain over raw Document objects is provided for cases where direct question–context *stuff* prompting is preferred.

Development Note: In 2.2.4.3 the retriever is explicitly configured with fetch_k, k, and lambda_mult, and the output is deduplicated and source-tagged, improving stability and traceability over earlier versions.

Library / Module	**Role**	**Limitation**
pathlib.Path, sys	Project root resolution and import path injection for configuration.	Path mutations can hide import issues if structure changes.
warnings	Suppresses noisy retriever warnings.	May hide useful deprecation/runtime hints.
typing (List, Set)	Type hints for docs and de-dup sets.	No runtime enforcement.
config.llm, CHROMA_PATH, COLLECTION, NUM_RESULTS	Externalized LLM client and RAG settings.	Misconfiguration breaks retrieval; static NUM_RESULTS not adaptive.
langchain.schema.Document	Document container for retriever/QA chain.	Minimal metadata schema; no ontology hooks.
langchain_openai.OpenAIEmbeddings	Embedding function for the vector store.	External API cost/latency; model/version dependence.
langchain_chroma.Chroma	Vector store client with persistent directory.	Local-only persistence; limited horizontal scaling.
chromadb.config.Settings	Chroma configuration (persist dir, telemetry).	Static, environment-sensitive paths.
Chroma.as_retriever(search_type="mmr")	MMR retriever to balance relevance/diversity.	Heuristic; may omit niche but critical chunks.
langchain_openai.ChatOpenAI, langchain.chains.question_answering.load_qa_chain	Optional QA chain (*stuff*) over raw docs.	Token-heavy; black box prompts; not default path here.

Table 31 - Libraries and Limitations RAG v2.5

d) DeepSearch

This module performs iterative query refinement using only local context (seed text from PDFs), without altering RAG logic or calling the web. It normalizes the provided context, prompts the LLM to propose keywords and a one-line Refined query, and accepts the refinement only if it is heuristically “better” (more specific or longer). The loop stops early when no further improvement is detected, and it returns the final refined query plus human-readable notes. It caps the prompt context (MAX_CTX_CHARS=4000) and degrades gracefully if the LLM is unavailable, in which case it simply skips refinement.

Development Note: New in 2.2.4.3, the module adds structured note generation ("Keywords:" / "Refined:"), early-stop heuristics, and optional language preservation via upstream flags, improving reproducibility and transparency of refinement decisions.

Library / Module	Role	Limitation
from __future__ import annotations	Enables postponed evaluation of type annotations for cleaner signatures.	No runtime impact beyond typing; Python version specific semantics.
typing (Any, List, Tuple)	Type hints for context normalization and return values.	Hints only; no runtime enforcement.
re	Extracts "Keywords:" and "Refined:" lines; simple pattern parsing.	Fragile to prompt/LLM formatting changes; local sensitivity.
config.llm (try/except import)	LLM used to propose refined queries from local context.	External API dependency; when absent, module skips refinement and returns fallback notes.

Table 32 - Libraries and Limitations DeepSearch v2.5

e) WebSearch

This module performs site-specific web retrieval and lightweight scraping. It queries DuckDuckGo via the ddgs client, filters results to a whitelist of trusted domains, and optionally falls back to unfiltered links if no trusted hits are found. For selected URLs, it fetches the page with a custom user agent and extracts readable text from paragraphs, list items, and headings, returning concise plain text for downstream summarization and prompt composition.

Development Note: In 2.2.4.3 this node is fully operational (not a placeholder). It adds explicit trusted domain filtering, a controlled fallback mode, bounded scraping with timeouts, and a consistent extraction policy for stable inputs to the summarizer.

Library / Module	Role	Limitation
ddgs.DDGS	DuckDuckGo client to retrieve text results for a query.	Third-party service; result shape may change; throttling possible.
urllib.parse.urlparse	Extracts netloc to enforce the trusted domain whitelist.	Simple string check; redirects/CDNs can evade strict filtering.
requests	HTTP GET with custom headers and timeout for page fetch.	No JS rendering; dynamic content not captured; timeouts/errors to handle.
bs4.BeautifulSoup	Parses HTML and extracts text from p, li, h2, h3 (bounded).	Ignores scripts/dynamic DOM; limited tag set; truncates to first N items.

config.USER_AGENT	Sends an explicit UserAgent to reduce blocks and improve parity.	Must be maintained; misconfiguration can lead to 403s.
config.TOP_K_URLS	Caps number of search results requested.	Static cap; may miss relevant items if too low.
config.TRUSTED_DOMAINS	Whitelist controlling which domains are accepted by default.	Requires curation; overly strict lists reduce recall.
run_web_search()	Orchestrates DDG query, filtering, and fallback policy.	If fallback enabled, may return off-domain links that need postfiltering.
scrape_page_text()	Fetches and distills readable text from a page.	Returns empty on fetch errors; content quality depends on page HTML.

Table 33 - Libraries and Limitations Search Web v2.5

f) DeepSearch Online

The *DeepSearch Online* module expands retrieval capabilities beyond the local indexed standards. It performs iterative search across external online sources, reformulating queries where necessary to increase retrieval diversity and depth. The module integrates results with the RAG pipeline, providing extended knowledge coverage when local context is insufficient. This enables the agent to address complex or multi-layered user queries with broader domain information.

Development Note: This version introduces structured query reformulation and multi-pass search depth, an improvement over earlier iterations which only returned shallow web results. Results from this module are now better ranked and seamlessly integrated into prompt composition.

Library / Module	**Role**	**Limitation**
duckduckgo_search.DDGS	Executes web queries on DuckDuckGo search API.	Coverage limited to DuckDuckGo index; variability in results.
tenacity	Retry logic for robust search attempts.	Adds latency; cannot fix upstream API unavailability.
bs4.BeautifulSoup	HTML parsing and text extraction from result pages.	Sensitive to site structure changes.
requests	HTTP requests for retrieving web pages.	Dependent on network; rate limiting from hosts.
re (regex)	Cleans and extracts relevant text from HTML.	May miss context; brittle on malformed HTML.
time	Adds delay between requests to avoid blocking.	Slows down query-response time.
json	Handles serialization of search results.	Limited to structured data; not semantic.

sys, os	System-level integration, environment handling.	Minimal role; mostly auxiliary.

Table 34 - Libraries and Limitations DeepSearch Online v2.5

g) Conversational LOIN

This module implements the conversational memory and state tracking for the prototype. It manages user responses against a predefined set of LOIN fields, ensures completeness, and provides contextual summaries for prompt composition. Its role is to keep dialogue structured and progressively build a record of information aligned with the LOIN specification.

Development Note: In version 2.2.4.3, LOIN conversation tracks not only responses but also the last-asked field, making turn management more consistent. Compared to earlier iterations, it provides a more reliable framework for multi-turn interactions, though still limited to in-memory state without persistence across sessions.

Library / Module	**Role**	**Limitation**
(custom logic only)	Session handling, turn management, and field tracking.	No external persistence; conversation state lost if session ends.

Table 35 - Libraries and Limitations Conversational LOIN v2.5

h) Convo Agent

This node provides the guided LOIN flow: detects user intent to request a LOIN deliverable, starts a short, non-specialist Q&A, loads the canonical ISO7817 field order from a CSV template, and then infers field values by fusing signals from history, local RAG context, optional web snippets, and optional ontology hints. It also formats a structured block ([LOIN structured context]) for downstream summarization.

Development Note: New in 2.2.4.3 are the CSV-driven field schema, optional ontology term pool loading, and a robust JSON extractor for LLM outputs. These additions make the LOIN path configurable and more fault-tolerant.

Library / Module	**Role**	**Limitation**
typing (List, Tuple, Optional, Dict, Any)	Type hints for public API and helpers.	Hints only; no runtime enforcement.
os, json	File/path checks (TTL), lightweight JSON parsing of LLM output.	No schema validation; JSON fixes are heuristic (_safe_json_load).
app.converational_loin.LOINConversation	In-memory store for field prompts/answers, completeness. checks	No persistence; state lost between runs.
app.csv_utils.load_table_structure	Loads canonical LOIN fields from CSV (order + labels).	Depends on CSV correctness; breaks if path/format changes.

config (LOIN_CSV_PATH, ONTOLOGY_ENABLED, ONTOLOGY_TTL_PATH, CONVERSATIONAL_LOIN_ENABLED, llm)	Feature flags, file paths, and LLM client binding for inference.	Static at startup; missing files/flags disable features silently.
rdflib.Graph *(lazy import; optional)*	Parses ontology TTL to build a term-hint pool for inference.	Only used if ONTOLOGY_ENABLED; parsing can be slow/fragile on large TTLs.

Table 36 - Libraries and Limitations Convo Agent v2.5

i) LOIN Schema

This module declares the canonical field order used to author and render LOIN deliverables. It provides a single source of truth for the sequence of keys expected in the conversational flow, inference, and structured output blocks. By centralizing the order, downstream components (e.g., convo_agent.infer_loin_fields() and build_structured_block()) can ensure consistent table layout and stable serialization independent of runtime conditions.

Development Note: In 2.2.4.3 the schema is expressed as a static constant list (CANONICAL_ORDER) reflecting the ISO-aligned headings used throughout the LOIN pathway. This replaces scattered, in-code lists found in earlier iterations and ensures consistent ordering between inference and summarization.

Library / Module	**Role**	**Limitation**
CANONICAL_ORDER (module constant)	Defines the authoritative, ordered list of LOIN fields consumed by conversational inference and structured rendering.	Static list; changes require code edit and redeploy; no runtime validation or localization switching.

Table 37 - Libraries and Limitations Loin Schema v2.5

j) CSV Utils

Utility module to locate, read, and normalize LOIN tables from CSV/XLSX. It resolves a path or directory to a recognized filename, loads tabular data with robust encoding fallbacks, normalizes headers to a canonical LOIN field order, and returns both the (field, description) pairs and the raw DataFrame for downstream use. Supports both long form (Field/Description columns) and wide form (headers as fields).

Development Note: Adds header normalization via aliasing and Unicode de-accenting, plus ordering against a single CANONICAL_ORDER, ensuring consistent field sequencing across ingestion sources.

Library / Module	Role	Limitation
os	Path expansion, dir/file checks, filename resolution from common candidates.	OS-dependent behavior; no advanced error context on permissions.
typing (List, Tuple)	Type hints for function interfaces and returns.	Hints only; no runtime validation/enforcement.
pandas as pd	Loads CSV/XLSX (with encoding fallbacks), column ops, NA handling.	Adds heavy dependency; Excel support requires engine; large files may be slow.
unicodedata	Unicode NFKD normalization and accent stripping for header aliasing.	Heuristic normalization; may oversimplify non-Latin scripts.
COMMON_FILENAMES (module const)	Filename shortlist to autodiscover table files in a directory.	Fixed list; misses unconventional filenames unless updated.
CANONICAL_ORDER (module const)	Authoritative ordered list of LOIN fields for column reordering.	Static; changes require code edit/redeploy; no local switching.
ALIASES (module const)	Header alias map (e.g., “why”→“Purpose”, “wbs”→“Breakdown structure”).	Heuristic; may need maintenance for new synonyms/labels.
_read_any_table()	CSV/XLSX loader with encoding retries and sep autodetect.	Fallbacks can mask data issues; engine differences across environments.
_map_headers_to_canonical()	Normalizes/renames headers and orders columns to canonical sequence.	Drops unrecognized structure from ordering; relies on alias coverage.
load_table_structure()	Public API: returns [(field, desc)] and the cleaned DataFrame (long or wide).	Raises FileNotFoundError if not located; minimal schema checking beyond headers.

Table 38 - Libraries and Limitations CSV Utils v2.5

k) Summarize

This node provides two summarization paths: summarize_simple for a direct BIM/RAG answer to the *original* question and summarize_loin for a LOINspecialized output that can include an ISO7817 table when sufficient evidence or a [LOIN structured context] is present. Both paths preserve the user’s language, avoid scope drift, and restrict themselves to provided context; summarize_loin also appends concise sections for Assumptions, Unresolved items, and Next Steps.

Development Note: This version introduces a split LLM strategy—llm_fast for responsiveness and llm_writer for higher quality LOIN specs/tables—and bakes in an exact canonical key order for table generation to keep outputs consistent across runs.

Library / Module	Role	Limitation
typing.List	Type hints for list-based inputs (external summaries, history).	Hints only; no runtime checking.
config.llm_fast	Fast LLM client for quick, plain BIM/RAG answers (summarize_simple) and fallback.	External API dependency; quality varies with model; rate limits/costs.
config.llm_writer	Higher quality LLM client used by summarize_loin for spec/table outputs.	Slower and costlier; still non-deterministic; requires network.
app.prompt_composer.compose_final_prompt	Assembles the base prompt from user query, RAG context, web summaries, and history.	No token budgeting here; relies on upstream inputs being filtered/concise.
CANONICAL_ORDER (module constant)	Exact ISO-aligned key order enforced when rendering the LOIN table.	Static list in code; edits require redeploy; "Not specified" used when evidence is missing.

Table 39 - Libraries and Limitations Summarize v2.5

l) Prompt Composer

This module is responsible for constructing prompts that guide the interaction between the prototype system and large language models. It aggregates retrieved chunks, user queries, and context metadata into a structured textual format, ensuring that downstream LLM calls receive coherent and standards-aware input. By defining consistent roles (system, user, assistant) and applying formatting rules, it enables more accurate and standards-aligned outputs.

Development Note: Compared to earlier versions, prompt_composer.py now integrates tighter coupling with RAG outputs and metadata injection from document schemas, improving alignment with ISO 23387 and ISO 7817-1 requirements for structured information delivery.

Library / Module	Role	Limitation
langchain.prompts (PromptTemplate, etc.)	Prompt construction utilities.	Templates can be rigid; require manual updates for schema evolution.
langchain.schema (SystemMessage, HumanMessage, AIMessage)	Defines structured conversational roles.	Limited semantic metadata; only text-based context is retained.
typing (Optional, List, Dict)	Type hinting for structured inputs/outputs.	Static typing only; no runtime enforcement.
os, json	Environment and serialization utilities.	No schema validation for JSON beyond manual checks.
(custom logic)	Composing multi-source context into unified prompts.	No automatic alignment with future schema extensions (manual maintenance).

Table 40 - Libraries and Limitations Prompt Composer v2.5

m) Logger

This node appends each interaction to a daily plain-text log under logs/session_YYYY-MM-DD.txt. For every question/answer pair, it records the search depth, the question, a trimmed PDF context (first 2,000 characters), the joined web summaries, and the final response. Files are created on demand and appended per interaction to maintain a chronological trace of the session.

Library / Module	**Role**	**Limitation**
os	Ensures logs/ exists; builds path to daily file (session_YYYY-MM-DD.txt).	No file rotation beyond daily; no concurrency/locking; platform-specific paths.
datetime	Generates current date string for filename.	System clock-dependent; no timezone normalization.
log_interaction(...) (custom)	Appends structured trace: search depth, Q/A history, trimmed PDF context (2,000 chars), web summaries, response.	Plain text only; no JSON/structured schema; possible PII; no size cap; errors not centrally handled.

Table 41 - Libraries and Limitations Logger v2.5

This page is intentionally left blank

4. PRELIMINARY EVALUATION AND RESULTS

This chapter aims to validate the feasibility and performance of the AI-driven strategies approach proposed in this research. By performing the research methodology outlined in Section 3.1 of the thesis, this phase focuses on the internal assessment of the developed prototypes.

All tests in this phase were conducted by the researcher. This decision was made to maintain full control over the testing parameters, input data, and evaluation metrics, thereby allowing for an accurate attribution of performance differences to specific architectural or functional changes implemented across the prototype iterations.

Given the exploratory and experimental nature of the prototype development process in which architectural strategies were incrementally applied, a preliminary evaluation was necessary to monitor their impact on performance. This also allowed for the early identification of functional trade-offs, such as improvements in precision at the cost of response stability.

The evaluation is therefore not intended to offer conclusive evidence of system superiority over standalone LLMs, but rather to serve as a structured benchmark for understanding the evolution of the prototype and assessing whether the proposed strategies support the effective operationalization of LOIN principles within BIM workflows.

These objectives are framed within the broader research aim of validating whether AI-driven assistants can operationalize LOIN in a structured, consistent, and verifiable manner, as defined by ISO 7817-1 and related standards. To this end, the evaluation focuses on three main research questions:

- To what extent do the prototypes improve the precision and consistency of standalone LLMs?
- How does the performance of the system evolve across successive architectural iterations?
- What is the measurable impact of individual strategies on the overall performance and stability of the system?

By addressing these questions, the evaluation aims to provide a structured basis for identifying the most effective design configurations and the best-performing prototype. It also supports the formulation of grounded hypotheses to guide future large-scale validations and collaborative testing phases.

4.1. Evaluation Parameters

The evaluation to perform consists of an Input Variable Test and a Conversational Flow Simulation, following the methodology described in Section 3.1. The Input Variable Test applies two fixed prompts to probe the formal definition of LOIN and the correct structuring of its three dimensions (geometric, alphanumeric, documentation). Meanwhile, the Conversational Flow Simulation replicates a realistic interaction where a non-expert user defines information requirements for a small design task; this evaluates terminology guidance, standards awareness, and adaptability when scope changes occur.

All models were evaluated with the same prompts and rubric. Table 42 lists the models (baselines and progressively refined prototype versions). Each entry specifies the LLM backend used during testing. The identifiers (M01–M11) are unique and used consistently throughout Chapter 4 for cross-reference.

Model Number	Model Name	LLM
M01	Ollama-deepseek	Deepseek-r1:latest
M02	OpenAI-ChatGPT	Gpt-5 Automatic[1]
M03	OpenAI-ChatGPT-CustomGPT	Gpt-5 Automatic
M04	LOINbot 0.0.6 (stable version 1.1)	Deepseek-r1:1.5b
M05	LOINbot 0.1.3 (stable version 1.2)	Gpt-3.5-turbo
M06	LOINbot 0.1.9 (stable version 1.3)	Gpt-3.5-turbo
M07	LOINbot 1.1.1 (stable version 2.1)	Gpt-4o-mini
M08	LOINbot 2.1.2 (stable version 2.2)	Gpt-4o
M09	LOINbot 2.2.2 (stable version 2.3)	Gpt-4o
M10	LOINbot 2.2.4 (stable version 2.4)	Gpt-4o
M11	LOINbot 2.2.4.3 (stable verions 2.5)	Gpt-4o

Table 42 - List of Models to Evaluate

4.1.1. Evaluation Metrics and Scoring

The performance of each answer given by the interactions is scored on six metrics: LOIN compliance; semantic/ontology compliance; completeness; precision; traceability; structure/legibility. Table 43 defines each metric and its evaluation goal with reference to ISO 7817-1, ISO 19650-1/-2, ISO 29481-1, and ISO 23387.

	Parameters	Description	Goal
A	LOIN Compliance	Does the answer respect the standards structure?	To assess the level of compliance with ISO 7817-1 and ISO 19650-1/2.
B	Semantic/Ontology Compliance	Is the answer consistent with the templates, terminology, and methodology of the standards?	To assess the semantic compliance with ISO 7817, ISO 19650-1/2, ISO 23387 and ISO 29481-1.
C	Completeness	Are all required attributes, geometry, and documents addressed?	To assess the sufficient amount of content in the interaction.
D	Precision	Is terminology correct, references consistent and not contradictory?	To assess the correctness of content and lack of hallucination in the interaction.
E	Traceability	Are the needs justified, linked, or auditable through rules or rationale?	To assess the use of the correct sources on the interaction.

[1] To evaluate the M02 the queries were input in a new user using a VPN to test the proficiency of the model itself.

F	Structure/Legibility	Is the answer presented clearly, organized, and easy to interpret?	To assess if the interaction is comprehensive as well as aligned with the ISO 7817, ISO 19650-1/2, ISO 23387 and ISO 29481-1 templates.

Table 43 - Evaluation Metrics

Each of the parameters are graded on a 0–5 scale as specified in Table 44, where "0" denotes failure and "5" denotes full compliance.

	Criteria	**Description**
0	Failure	No relevant output retrieved, or the content is entirely irrelevant to the declared requirement.
1	Incorrect	Any content is acceptable as long as it is "similar" to BIM, even if unrelated to the defined requirement. Reviewer judgment is optional and consistency is not necessary.
2	Partial	Significant problems: large gaps (20–40% of content missing or wrong), multiple contradictions, or lack of traceability. Output is only partially usable and requires substantial rework.
3	Acceptable	Moderate issues: noticeable omissions or errors (about 10–20% of content affected), but the output is still interpretable and usable with limited adjustment.
4	High Compliance	Minor deviations only: one or two small issues (e.g., slight formatting or terminology inconsistency). The output remains fully usable without correction.
5	Full Compliance	Output is fully aligned with requirements: complete, correct, unambiguous, directly usable in practice. No omissions, errors, or contradictions.

Table 44 - Evaluation Criteria

As stated in the introduction, all tests were executed by the researcher to ensure control over inputs, scoring, and environment. The same dataset and prompts were used across all models; each test was repeated across three independent sessions per model to gauge consistency. This setup supports fair comparison between standalone baselines and the prototype variants and enables attribution of differences to architectural changes rather than test variability.

4.2. Diagnostic Phase: Standalone Benchmark (M01-M03)

The first benchmark phase of the evaluation was designed to establish a baseline of performance for general-purpose LLMs operating without any ontology-based or retrieval-augmented enhancements. Three models were selected (M01–M03), each tested under identical conditions using the parameters, metrics, and scoring rubric defined in Section 4.1. This phase serves as a reference point for assessing the incremental improvements introduced in the subsequent prototype versions.

4.2.1. Test 1 – Input Variable Test

The first test measured the capacity of standalone LLMs to produce a precise, standards-compliant definition of the LOIN. Models were prompted with a direct query requiring reference to the concepts and structure described in ISO 7817-1, ISO 19650-1/2, and related standards according to the parameters defined in Section 3.1.2 Input Variable Test and detailed in Table 1.

The results presented in Figure 13 shows the average mean score for the Test 1 for M01, M02 and M03 across three runs. The figure highlights a clear performance difference across the three models. While DeepSeek-r1:latest1 and GPT-5 Automatic delivered only partial or generic responses, the Custom GPT version consistently achieved higher scores across all runs, averaging above 3.4. This indicates that while baseline models can approximate definitions, they often lack the necessary alignment with ISO 7817-1 and related standards, resulting in missing references or incomplete structuring of LOIN dimensions. In contrast, the Custom GPT demonstrated stronger recall of domain-specific terminology and better structural clarity, suggesting that prompt conditioning plays a role in improving baseline compliance.

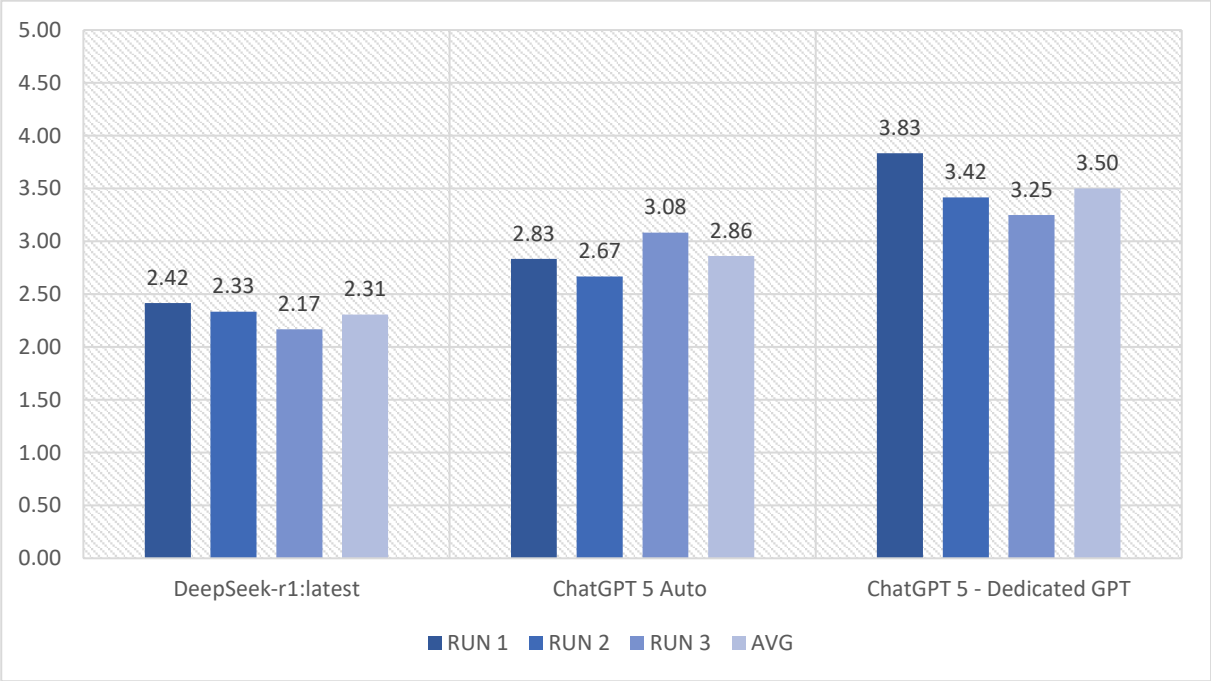


Figure 13 – Test 1 LLMs average mean score

4.2.2. Test 2 – Conversational Flow Simulation

The second test simulated a multi-turn conversational exchange in which a user incrementally defines information requirements. The objective was to evaluate not only the correctness of the responses but also the ability to interpret the information, maintain context, coherence, and consistency across the

[1] Deepseek-r1:latest correspond to the model r1:8b and it was tested in a local environment as it was the biggest model able to run in a standard hardware.

dialogue. The Test 2 is been perform according to the parameters defined in Section 3.1.3, and detailed in Table 2.

The results presented in Figure 14 shows the average mean score on Test 2 of M01, M02 and M03 across the three runs. The outcomes presented in the figure show a noticeable decline in performance compared to the static query of Test 1. Both DeepSeek-r1:latest and GPT-5 Automatic averaged below 1.8. Responses tended to lose context over the sessions, often defaulting to generic BIM explanations rather than maintaining focus on LOIN or standard-based structuring. The Custom GPT again outperformed the other models, averaging 3.5, and was the only one capable of maintaining a consistent dialogue aligned with the test objectives.

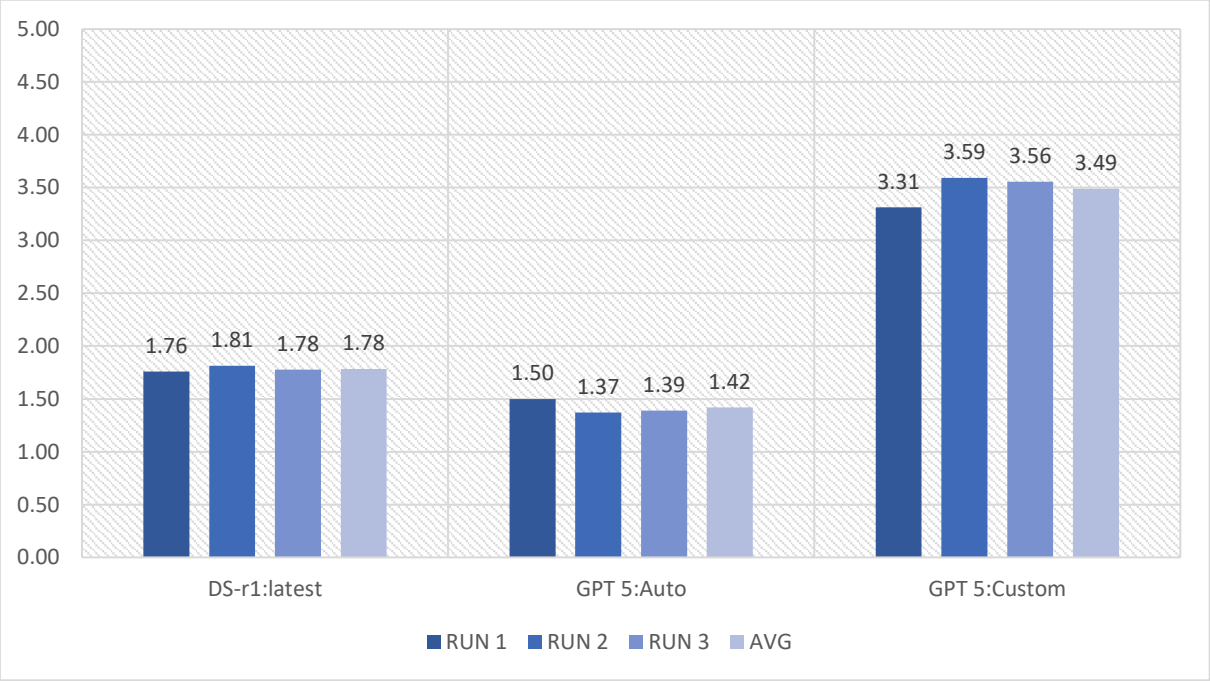


Figure 14 – Test 2 LLMs Average Mean Score

4.2.3. Parameter Compliance Across Tests

To complement the individual results of Test 1 and Test 2, the average performance of each model was also aggregated by the evaluation parameters (Table 43). This combined analysis provides an overview of systemic strengths and weaknesses over the evaluation criteria.

The benchmark evaluation of standalone LLMs demonstrates that, although these models are capable of producing outputs that appear complete and well-structured, their performance is inconsistent and fundamentally limited in relation to LOIN principles. Results from both tests show that weaknesses are systematic in precision and traceability, while partial strengths are observed in completeness and structural clarity. These findings establish a clear baseline of performance and highlight the need for domain-specific enhancements, retrieval augmentation, and ontology-based structuring in the prototype development to be analyzed.

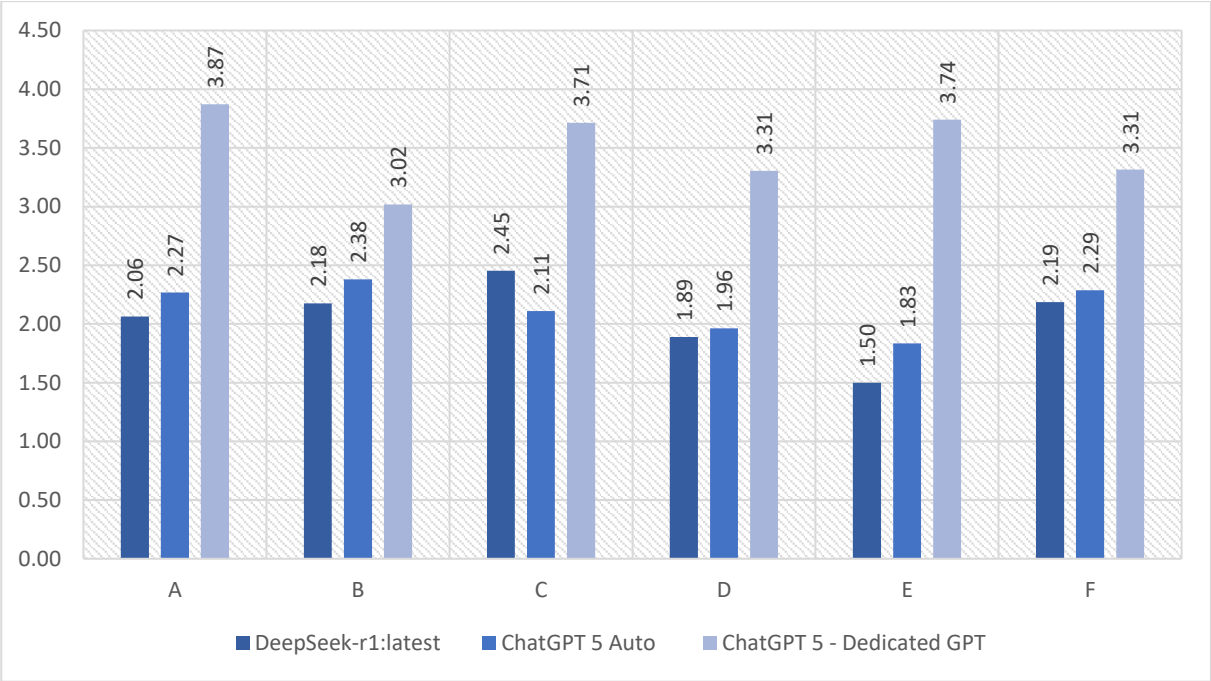


Figure 15 - LLMs aggregated average score across evaluation parameters

4.3. Guided Phase: Prototype Benchmark (M04-M10)

This second benchmark phase evaluated the sequence of prototypes (M04–M11) under the same parameter of Section 4.2 and defined on Section 3.1 of this thesis, each representing an iteration of architectural and functional strategies as described in Chapter 3. In contrast to the standalone LLMs presented in Section 4.2, the prototypes progressively incorporated RAG, refined chunking approaches, specialized prompt engineering, and partial ontology-based enhancements. The objective was to assess whether these strategies produced measurable improvements in LOIN compliance, semantic rigor, and conversational stability.

4.3.1. Test 1 – Input Variable Test

The results shown in Figures 16 and 17 present the evolution of prototype performance and the comparative outcomes across all models. Early prototypes (M04–M06) underperformed, with average scores close to or below 1.0, indicating that the first implementations of RAG and chunking strategies were insufficient to reliably deliver standards-aligned definitions. From Model 7 onwards, performance improved significantly, reflecting the impact of refined implementation of RAG and chunking strategies. Models 9 and 10 achieved the best results, averaging close to 4.0 and delivering responses that were consistently aligned with ISO 7817-1 definitions, Figures 17 and 18 show that the prototypes not only improved their average score but also their result consistency, with minimal variation between test runs.

M11, however, showed a marked decline in stability: across its different configurations, the average score dropped back to ~1.0. This indicates that the changes applied to assess the functionalities as well as the partial integration of ontology functions introduced instability into otherwise consolidated improvements.

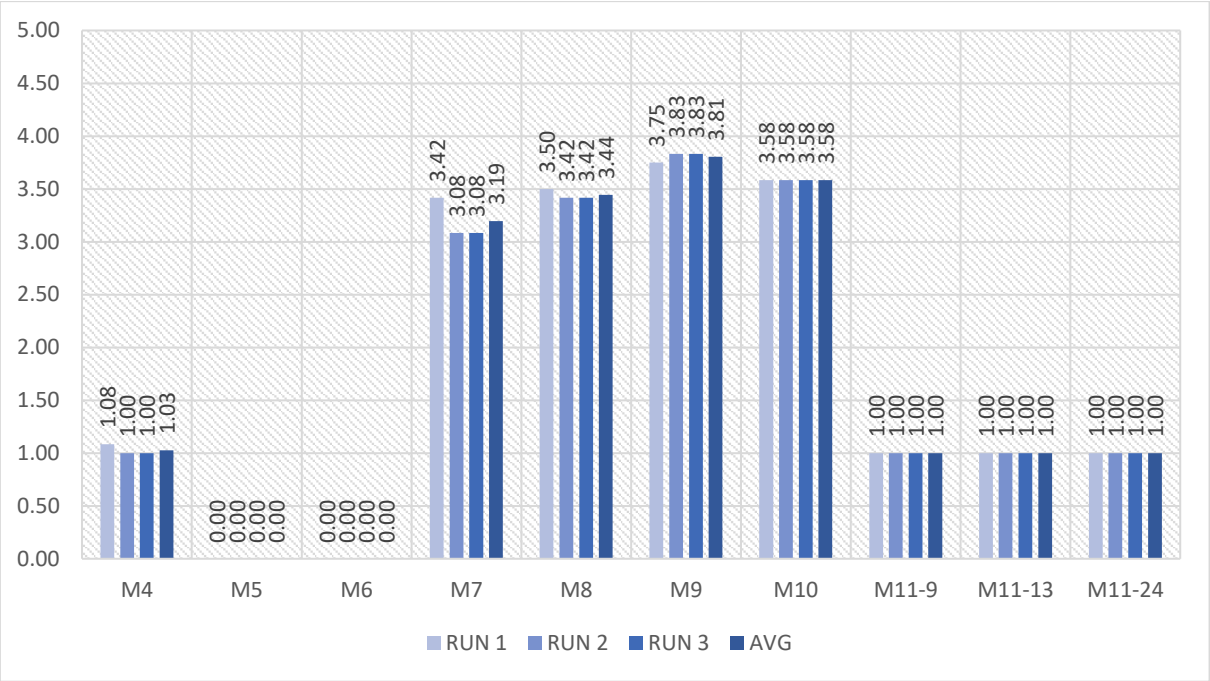


Figure 16 –Average mean scores of Test 1 across prototype development

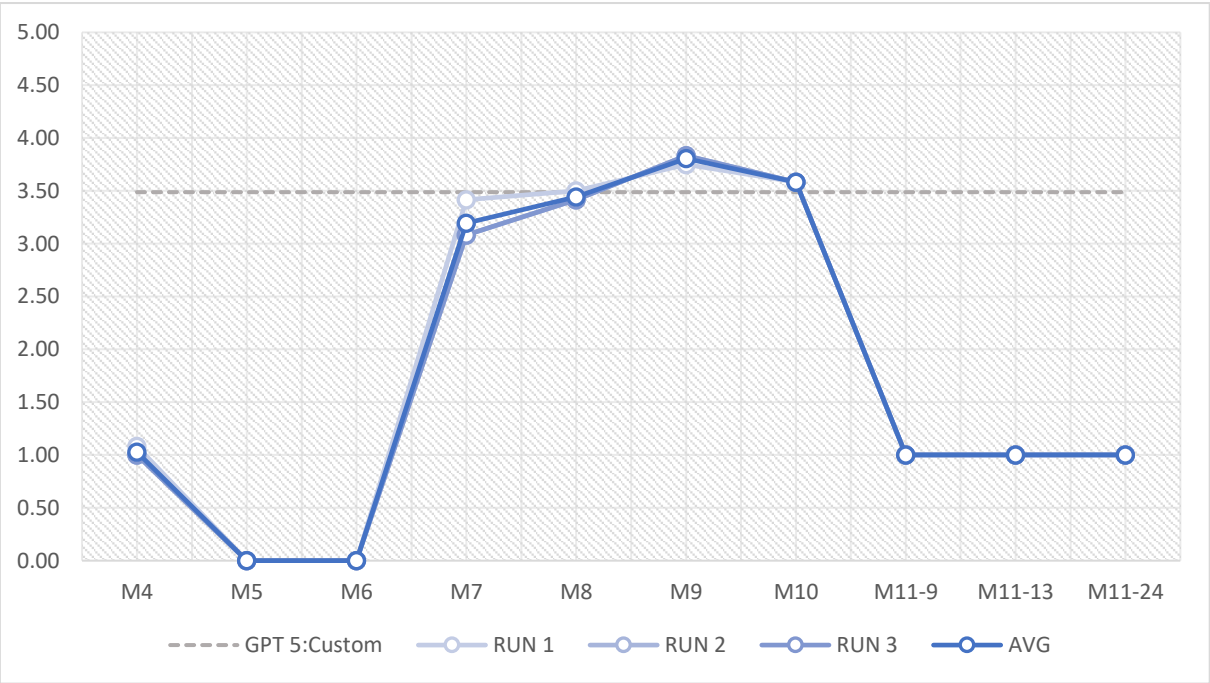


Figure 17 –Average scores evolution on Test 1

4.3.2. Test 2 – Conversational Flow Simulation

The outcomes presented in Figures 18 and 19 reveal a clearer trajectory of improvement. Once again, early prototypes (M04–M06) failed to maintain meaningful conversational flow, with averages below 1.0. Beginning with M07, performance improved steadily, peaking with M09 and M10, which consistently achieved averages above 4.6. These versions demonstrated strong contextual retention and produced responses aligned with the three LOIN dimensions across multiple conversational turns.

As with Test 1, consistency across runs was a defining improvement of M09 and M10. The small variation between runs demonstrates that the retrieval and prompt strategies introduced in these versions not only improved precision but also stabilized performance.

M11 presented mixed results. Although some configurations achieved averages above 4.0, performance was inconsistent across runs and sub-versions. The instability could be attributed to the change of approach and Boolean parameterization, which disrupted the integrity of the retrieval process and reduced repeatability compared previous prototypes. The future development and testing of the functionalities running in M11 could allow integration of ontology into the pipeline.

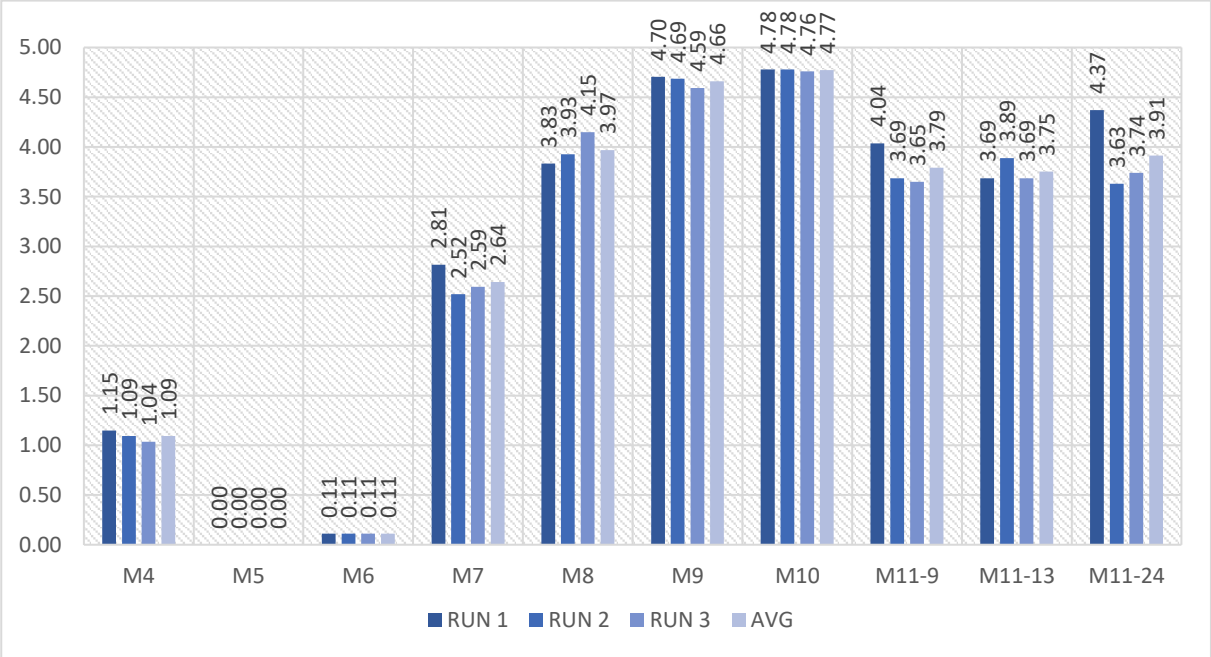


Figure 18 – Average mean scores of Test 2 across prototype development

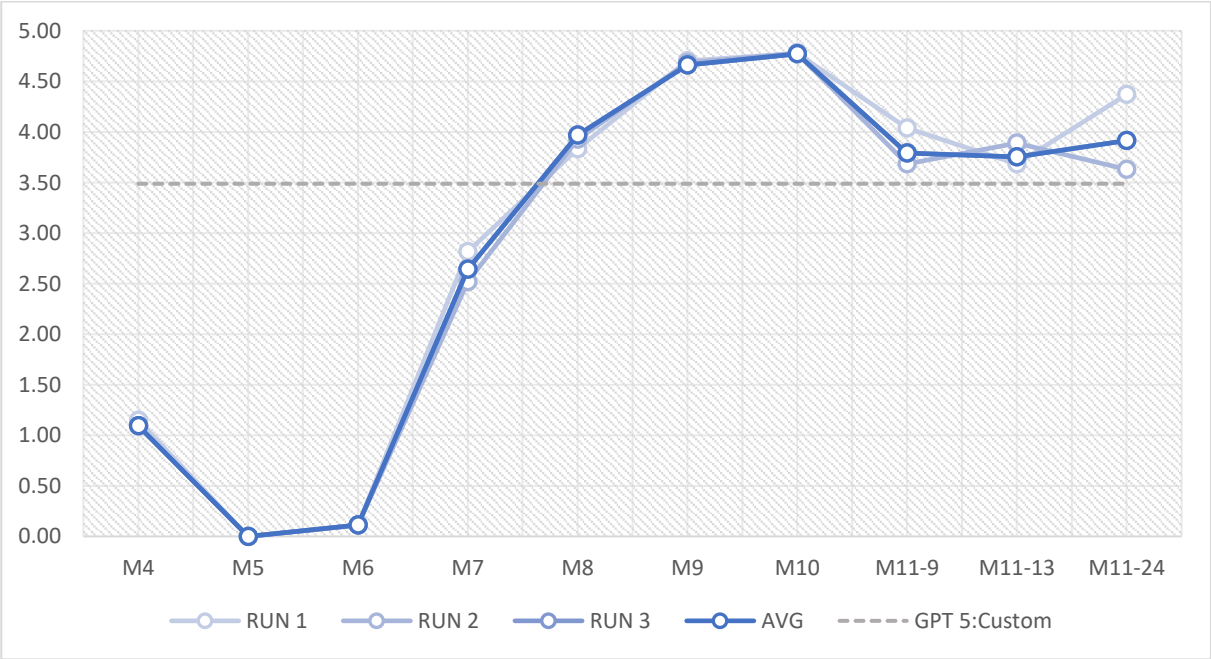


Figure 19 – Average score evolution on Test 2

4.3.3. Parameter Compliance Across Tests

Beyond model-to-model comparisons, parameter-level analysis highlights where prototypes improved most. Figure 20 compares average scores for M09, M10, and M11-24 across the six evaluation parameters. M09 and M10 achieved consistently high performance across the evaluation parameters (Table 43) A**,** B**,** D**,** and F, with scores above 4.0 in nearly all categories. Parameter E (traceability) remained a relative weakness, with averages around 3.7 for M09 and M10, and dropping sharply for M11.

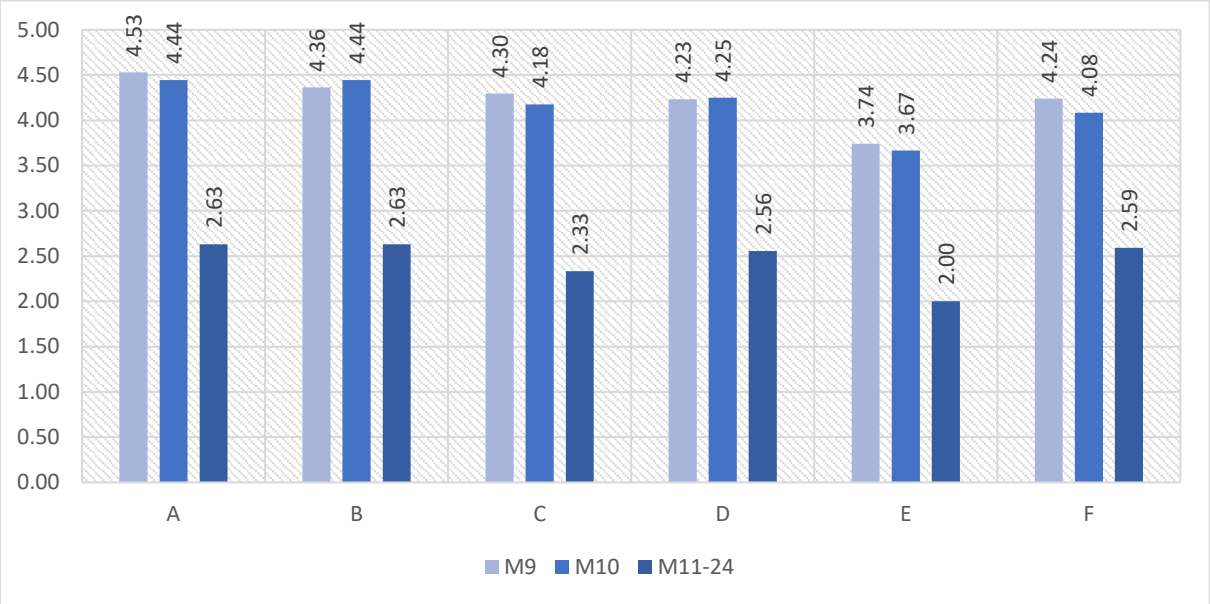


Figure 20 – Average prototypes score by parameter

The benchmark evaluation of prototypes shows a strong upward trajectory from M07 through M10, with these versions clearly surpassing standalone LLMs in both definition accuracy and conversational stability. The most effective strategies were refined retrieval and chunking approaches, supported by prompt engineering. However, the decline observed in Model 11 underscores the risks of altering retrieval logic through Boolean controls before stabilizing the core architecture. Although ontology integration was introduced in this version, it cannot be singled out as the cause of the instability. Instead, the results highlight the need for careful calibration of functional switches to preserve the integrity of prior improvements.

4.4. Evaluation of Improving Strategies (M11 configurations)

This subsection evaluates the current prototype M11, which introduced a configurable architecture through six functional variables. The purpose of this exploratory assessment is to examine how individual functions and their combinations affect the performance, stability, and output quality of the RAG agent.

4.4.1. Experimental Setup

The evaluation baseline of the different configurations of M11 is defined by the Configuration 1 (C1), where all functionalities were disabled, leaving the system to rely solely on the RAG and chunking strategies of this prototype. M11 exposes six Boolean functions that can be toggled to shape the agent's behavior during retrieval and response generation. Table 45 summarizes each switch, its label, and the intended effect on the pipeline.

	Function	**Description**
D	DeepSearch	Enables local DeepSearch to strengthen RAG.
R	Query Rewrite	Enables rewriting of the user query over local DeepSearch iterations.
C	Conversational LOIN	Enables a guided, assist flow aligned with LOIN requirements.
O	Ontology Alignment	Enables ontology terms for normalization, if TTL is available.
W	Web Search	Enables live web search on predefined domains.
WD	Web DeepSearch	Enables iterative web search.

Table 45 – M11 functional options

It is important to note that some functions are conditional (R requires D, WD requires W, and O requires an active ontology context), which limits the space to 27 valid combinations out of 64 theoretical ones; these 27 configurations constitute the evaluated set. Table 46 presents the configuration matrix (T = enabled, F = disabled).

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**
D	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
R	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	T
C	F	F	F	T	T	T	T	T	T	F	F	F	T	T	T	T	T	T	F	F	F	T	T	T	T	T	T
O	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T	F	F	F	F	F	F	T	T	T
W	F	T	T	F	T	T	F	T	T	F	T	T	F	T	T	F	T	T	F	T	T	F	T	T	F	T	T
WD	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T	F	F	T

Table 46 - M11 list of configurations

Each configuration was run three independent times using the Conversational Flow Simulation (Test 2) under identical prompts defined in Table 2, dataset, and scoring rubric defined in Section 4.1. For every configuration, two summary statistics were computed: the mean score (0–5 scale across the six evaluation parameters) and the standard deviation across runs, to capture typical performance and run-to-run consistency.

4.4.2. Results Overview

The overall distribution of results presented in Figure 22 shows the three runs of each configuration together with their average score. The figures show that individual runs performance score ranged between 3.2 and 4.5 across configurations, meaning that some configurations remained competitive compared to earlier prototypes (M04–M08) but fall from the results exhibit by M09 and M10,

respectively. It's important to note that 11 out of the 27 configurations performed worst in average than the benchmark for this overview (C1), which scored 3.66, below M08's score of 3.97.

Figure 21 also reveals a visible increase in instability. While several configurations remained tightly around their averages, others showed large variation between runs. Notably, the best-performing configuration, Configuration 24 (C24) that scored an average of 3.91, was also among the least consistent. From these results, it can be concluded that in the M11 environment, although the application of the functionalities generates higher performance, this does not necessarily tie with higher stability**.**

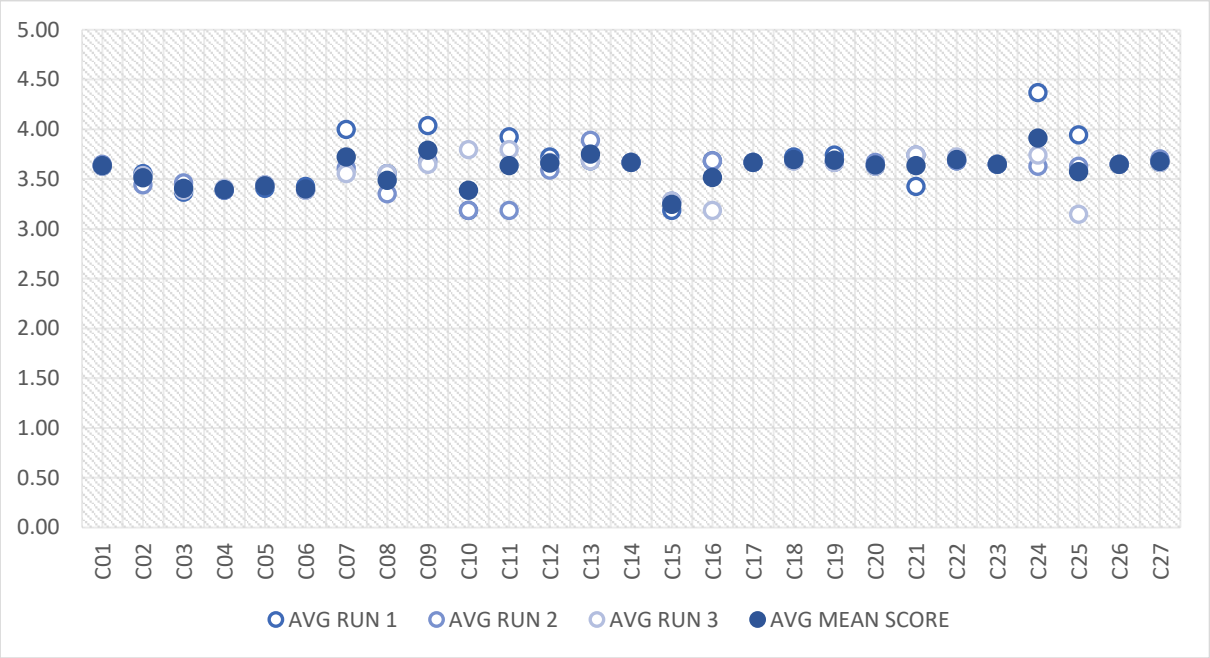


Figure 21 – Average mean scores of M11 scores across configurations

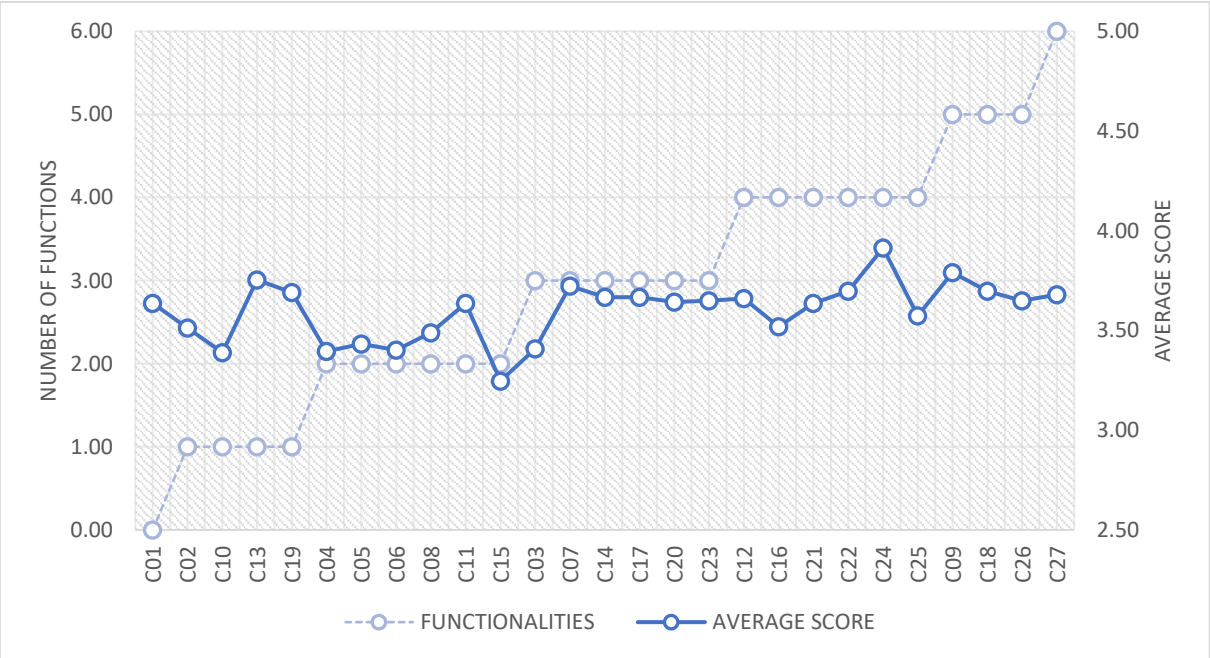


Figure 22 – Prototypes average performance across number of functions activated

In a second instance, the analysis also indicates no correlation between the number of active functions and overall performance, as reflected in Figure 22. Some configurations with only two or three functions enabled outperformed more complex setups, suggesting that aggregation does not guarantee linear improvement.

4.4.3. Stability Analysis

In order to search for the source of the drop in stability identified in Figure 23, the standard deviation was calculated for each configuration (Appendix 2). In a first analysis, the percentage of deviation was compared to the number of functionalities applied to M11. The results reflected in Figure 23 show no direct relationship between the number of functionalities running and the deviation on stability.

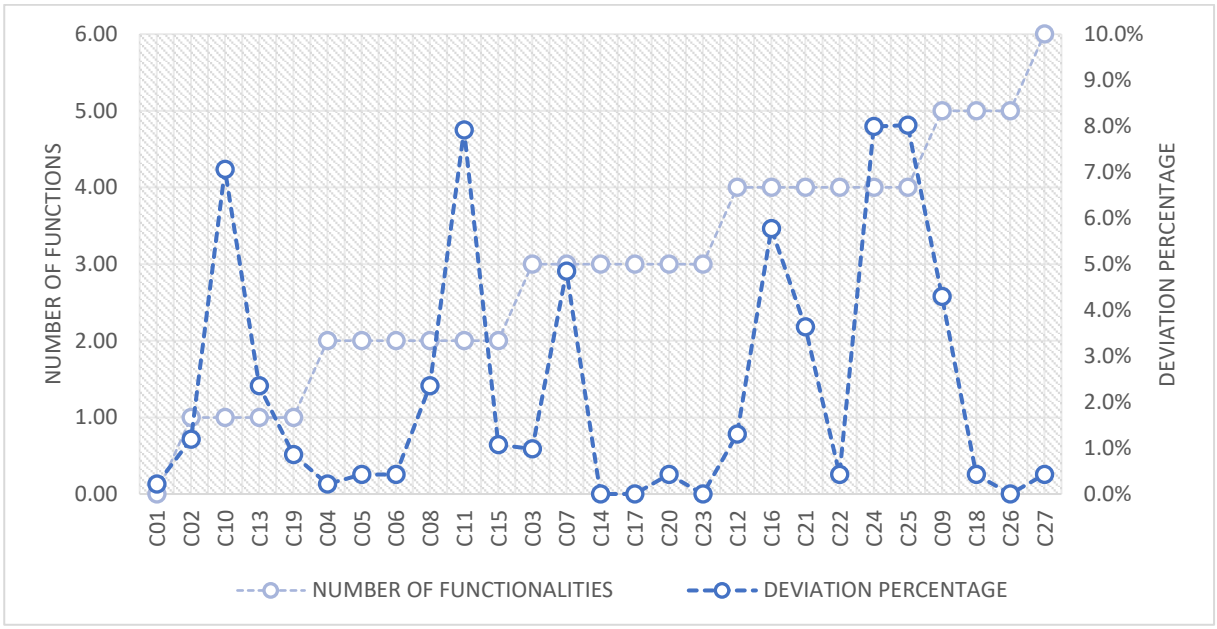


Figure 23 – Deviation percentage across number of functionalities applied

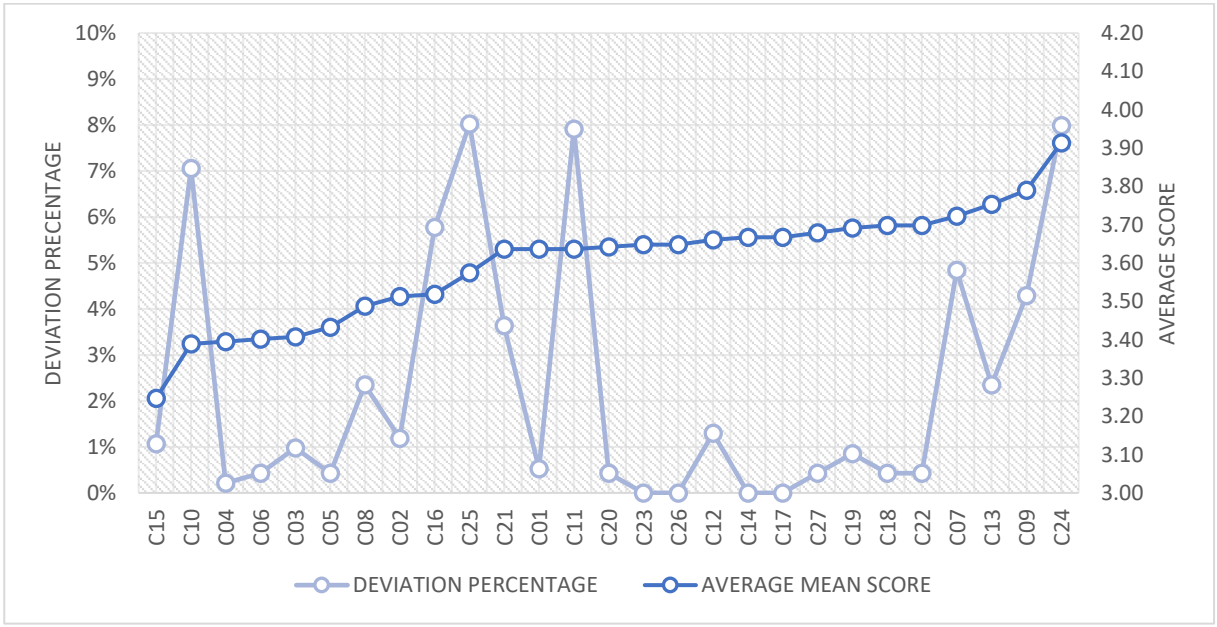


Figure 24 – Deviation percentage across average configurations score

From the results exhibited in Figure 23, 5 of the 27 configurations presented a deviation on the scores over 5%. From this sample, the configurations with the highest instability were C11, C24, and C25, each with an average deviation of 0.4 across runs. Interestingly, C24 also recorded the highest average score of 3.91, suggesting that peak performance was obtained at the cost of consistency. As reflected in Figure 24, other high-scoring configurations achieved more moderate deviations, highlighting that instability is not tied to a specific function load but emerges from particular combinations.

In a deeper analysis, the functionalities applied to the 5 least stable configuration (C07, C10, C11, C16, C24, and C25) and the functionalities running in the most stable configurations (C01, C04, C14, C17, C23, and C26) were compared to identify relations between the functions applied and the stability of the configurations. As reflected in Figure 25, besides function O, the usage of the different functionalities remained consistent, confirming the hypothesis that the decrease on stability derives from the impact of specific combinations on the tool's process logic. It is interesting to note that both samples ran a total of 18 modules.

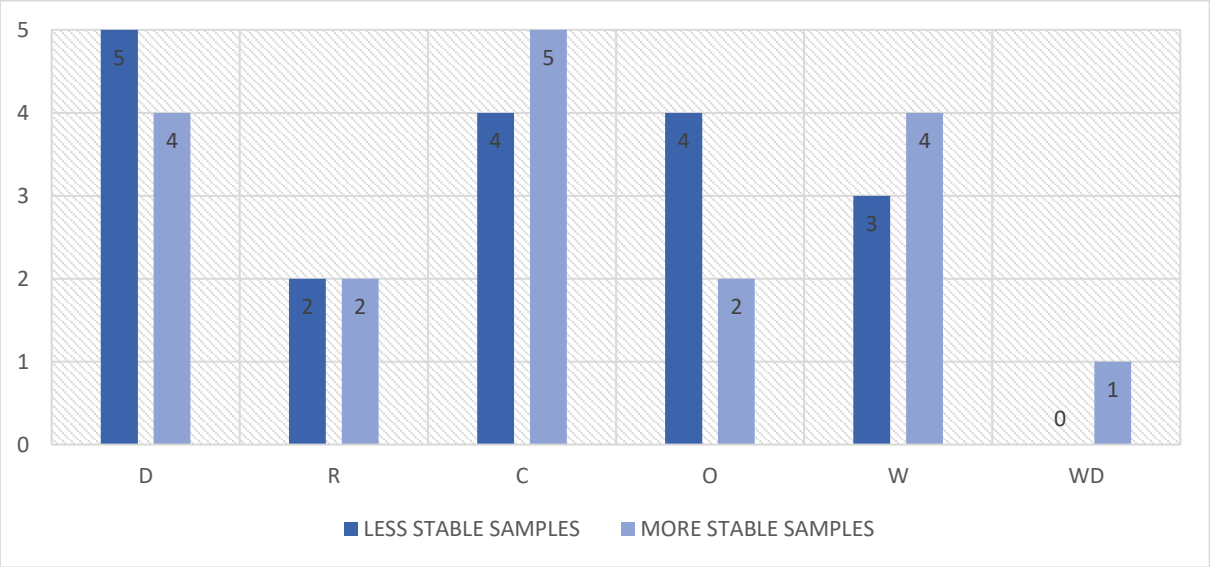


Figure 25 - Samples individual functionality usage

The results indicate that modular switching did not yield a systematic uplift: 11 of the 27 configurations performed below the C1 baseline, and observed gains were marginal rather than sustained. Because no correlation emerged between the number of enabled functions and mean performance, stacking features did not produce monotonic improvement; in several instances, leaner configurations outperformed more complex ones.

Because run-to-run instability was pronounced in several cases apparent peaks should be treated as non-reliable advantages rather than durable improvements. Consequently, the findings are indicative rather than causal, and the evidence does not support attributing effects to individual functions in isolation.

Given that the observed variability aligns more with how functions are implemented and orchestrated on top of the RAG pipeline rather than with the mere presence of switches, future development should prioritize refining function design and integration before expanding the switch set. Attention should return to M11's foundations: the RAG and chunking strategies, since the baseline configuration (C1)

underperforms M08, which relied on these mechanisms alone. Taken together, these observations suggest a two-step path: first stabilize and optimize the baseline functions and then reintroduce the functional modules incrementally under stricter integration logic, supported by larger testing for each configuration and a planned grid of valid on/off combinations to separate individual effects from combination effects.

4.5. Results Comparison: Analysis of M03 vs M10

Across both tests, the prototypes surpass standalone LLMs in precision and consistency when responding to LOIN-related tasks. Performance improves progressively through the development path, consolidating around M10. In direct terms, later prototypes deliver higher average scores and tighter run-to-run dispersion than the best standalone baseline under identical prompts and scoring (Section 4.1). Regarding the measurable advantage of the best prototype over the best standalone model, the uplift is clear. However, effects attributable to the new functional switches introduced in M11 are not conclusive. The observed variability aligns more with how functions are integrated and orchestrated on top of the RAG pipeline than from the modularity per se.

4.5.1. Statistical Performance Comparison, M03 vs M10

To quantify uplift under identical conditions, the comparison is restricted to the best-performing standalone baseline (M03) and the best-performing prototype (M10) using the same prompts, dataset, and 0–5 scoring rubric. Observations are treated as paired at the per-prompt (and per-run, when applicable) level so that differences reflect model behavior rather than task variation.

In order to quantify the improvement across models and parameters, the mean score on a 0–5 scale is also expressed as a Compliance Index (CI), defined as CI=($\bar{s}$/5)x100%, with $\bar{s}$ being the average score obtained. Measured under identical prompts and rubric (Section 4.1), while M03 attains CI ≈ 69.8% (mean ≈ 3.488), the prototype M10 attains CI ≈ 94.7% (mean ≈ 4.735).

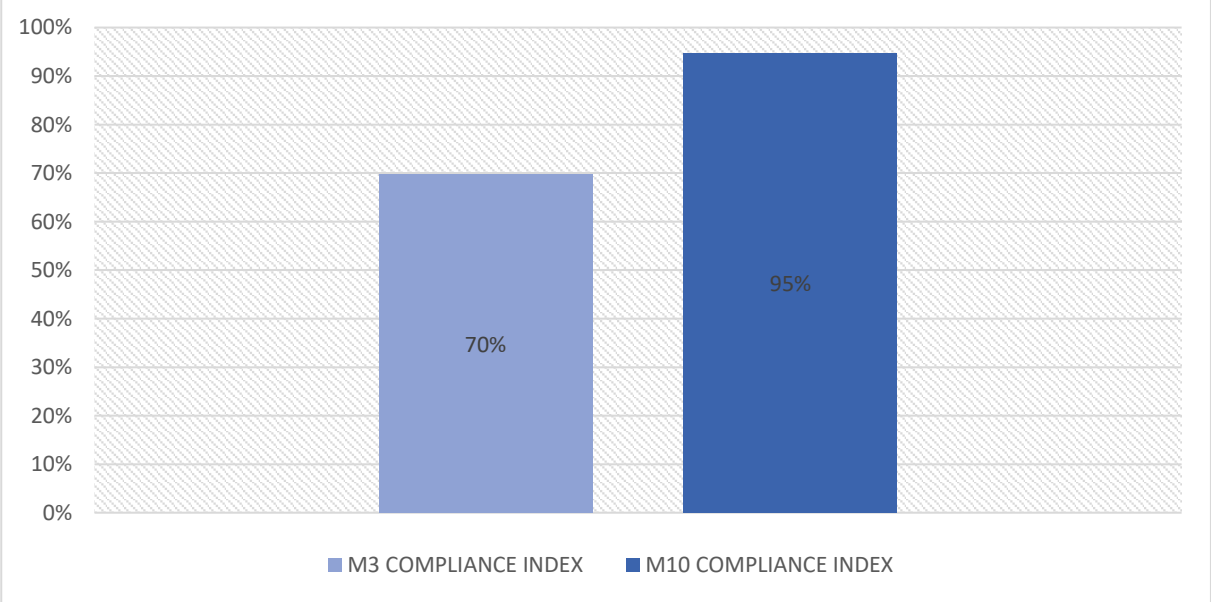


Figure 26 - Average performance compliance index of M03 and M10

As presented in Figure 27, across all prompts, M03 averaged ≈ 3.488 and M10 averaged ≈ 4.735, corresponding to an overall improvement of ≈ +35.75%. By parameter, the mean uplift was also positive in every case: LOIN compliance (A) +30.693%, semantic/ontology alignment (B) +60.976%, completeness (C) +35.052%, precision (D) +45.161%, traceability (E) +24.468%, and structure/legibility (F) +28.571%. These values indicate a broad-based advantage for M10 rather than a gain confined to a single metric.

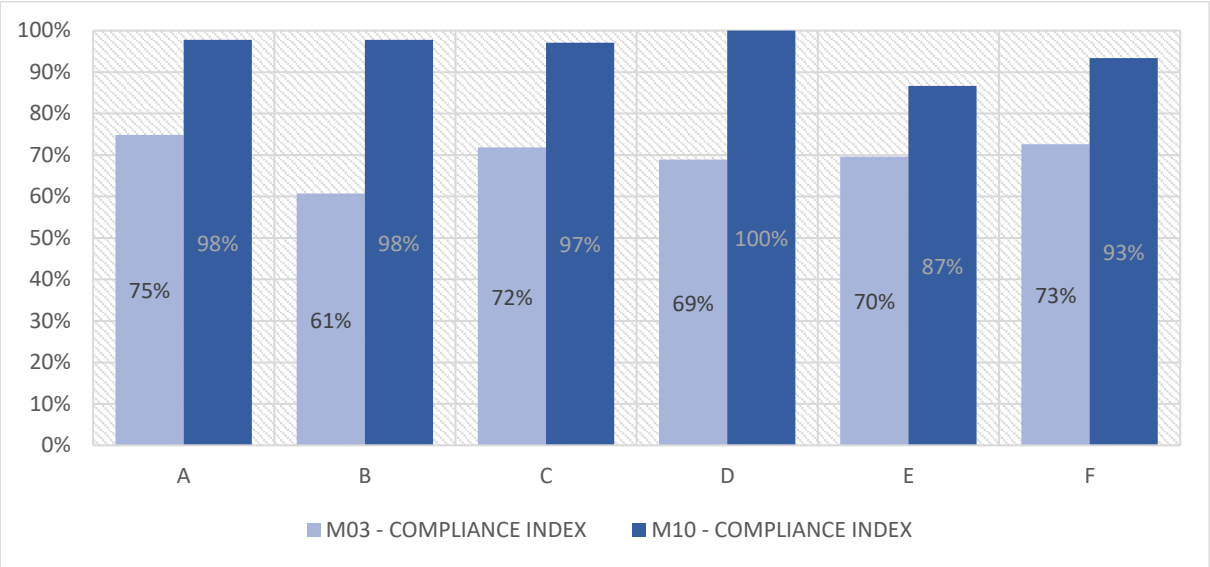


Figure 27 - Compliance index by Parameter of M03 and M10

4.5.2. Stability and Repeatability Comparison, M03 vs M10

Run-level dispersion favored M10. For parameters A, B, D, E, F, the run-to-run standard deviation (SD) for M10 was 0.000, implying no variation across its three runs; for C, M10's SD was 0.064. In contrast, M03's SDs were 0.292 (A), 0.139 (B), 0.277 (C), 0.091 (D), 0.105 (E), and 0.139 (F). Stability was assessed using the coefficient of variation CV=SD/Mean.

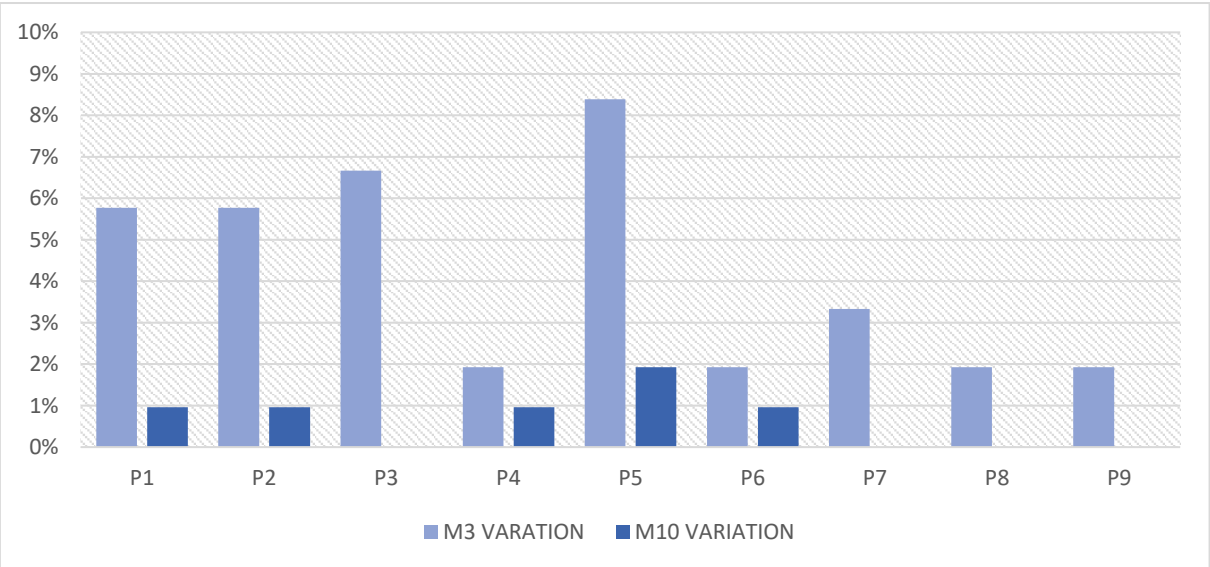


Figure 28 - Coefficient of variation by prompt between M03 and M10

As shown in Figure 28, the prototype's CV remained between ~0.9% and 2.0% across prompts, while the best baseline ranged between ~2% and 8.5%, with the largest gap in performance at P5. This indicates consistently higher repeatability for the prototype, normalized by its higher performance level.

M10 outperformed M03 on every prompt and every parameter across runs. This distribution of differences confirms that the advantage is general rather than driven by a small subset of prompts. Given that the prototype wins on all paired items, a simple direction-of-change check (sign test) would support the interpretation that the observed uplift is not due to random fluctuation. That said, the evaluation remains preliminary; the number of runs per configuration is limited, so these results are treated as indicative. A follow-up phase with more repeated runs per configuration and a planned grid of valid on/off combinations will allow stronger separation of individual effects from combination effects.

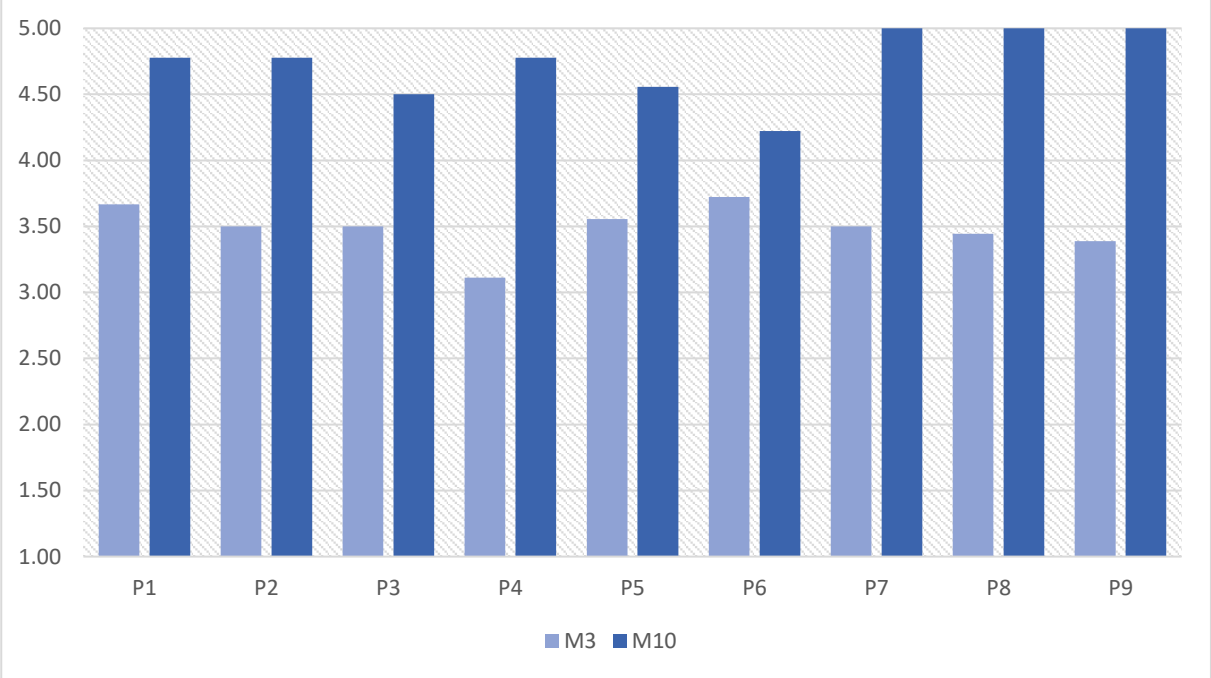


Figure 29 - Prompt average score comparison between M03 and M10

The implications of these results and the lessons derived from them are synthesized in Chapter 5.

5. CONCLUSIONS

5.1. Synthesis of Findings

This thesis examined whether an AI-driven, standards-aware assistant can support the definition and structuring of the LOIN in BIM workflows. The approach combined RAG over a curated corpus of standards with ontology-based normalization so that generated content would be grounded in shared terminology and expected document formats.

Measured under identical prompts (Section 3.1) and rubric (Section 4.1), the best prototype (M10) achieved a Compliance Index of approximately 94.7% (mean ≈ 4.735) versus 69.8% for the best standalone baseline (M03; mean ≈ 3.488). Improvements were positive across all parameters: LOIN compliance +30.693%, semantic/ontology alignment +60.976%, completeness +35.052%, precision +45.161%, traceability +24.468%, and structure/legibility +28.571%. Run-level dispersion favored M10. SD was 0.000 across runs for five parameters and 0.064 for completeness, with the coefficient of variation between ~0.9% and 2.0% across prompts, compared to ~2% to 8.5% for M03. Across prompts and parameters, M10 outperformed M03 on every paired item; a direction-of-change sign test would therefore support that the uplift is unlikely due to random fluctuation.

Taken together, these results of the preliminary evaluation suggest that standards-grounded automation is technically feasible and practically promising for authoring information requirements. With coherent passage boundaries, targeted prompts, and access to authoritative references, outputs were more structured, legible, and aligned with LOIN concepts than those from generic models. This effect was most evident in differentiating alphanumeric properties, geometric aspects, and supporting documentation, and in signaling missing contextual elements such as purpose, actors, or milestones.

Within the limits of the present sample, the strongest prototype consistently outperformed the baseline across repeated runs. The compliance index approached near-full alignment on the tested prompts. These figures are presented as indicators of direction rather than as definitive proof. They describe how targeted RAG, disciplined prompt composition, and ontology alignment can nudge generation toward higher consistency and traceability in the evaluated scenarios.

The underlying mechanisms appear to be architectural: RAG precision and chunking coherence supported better downstream reasoning, while uncoordinated feature additions did not yield monotonic gains and at times reduced stability. Ontology alignment promises benefits when vocabularies and templates are complete and consistently mapped; in heterogeneous areas the advantage diminishes and phrasing drifts toward the generic, but this cannot be confirmed within this thesis.

This pattern contrasts with the feature aggregation experiments of M11, showing no correlation between the number of active functions and mean performance (Fig. 22), and the highest-scoring configuration (C24, mean 3.91) was among the least consistent (average deviation ≈ 0.4), indicating that uncoordinated feature aggregation did not produce monotonic or stable gains.

5.2. Implications for BIM Practice

Despite the evaluation's limitations, the findings point to pragmatic opportunities for information management. The preliminary evaluation suggests that a standards-aware, AI-driven assistant can serve as a bridge between the user and the standardized knowledge defined by LOIN and related BIM standards, helping to articulate information needs with consistent terminology, semantics, formatting, and clear links to sources. These strengths map directly to ISO 7817-1's three LOIN components (geometrical, alphanumerical, documentation) and to ISO 19650-4's recommendation to use formalized, repeatable checks against information requirements, supporting verification workflows within a CDE.

Ontology, data dictionaries, and data templates could further strengthen performance by providing stable anchors for property names, units, and value lists. In the current state of the prototype, human validation remains essential to confirm alignment of requirements and manage risk in line with project governance.

5.3. Directions for Future Research

Subsequent work should broaden the evidence base and strengthen validity controls. Larger and more diverse evaluations, covering additional disciplines, project phases, languages, and regulatory contexts, would improve external validity. Having multiple independent reviewers score the same outputs and report agreements will help refine the rubric and verify that it measures what it is intended to constructs. Controlled experiments that isolate the effects of retrieval quality, chunking strategy, ontology alignment, and query rewriting would clarify impact and robustness and identify opportunities for improvement, building on the testing insights reported earlier in this thesis.

A key next step is to improve ontology support and to refine the ontology's composition so it aligns more closely with the agent's goals. Concretely, this entails expanding coverage for domain vocabularies and data templates referenced during generation, tightening mappings between classes/properties and LOIN dimensions, normalizing units and permissible values, and resolving heterogeneities across data dictionaries. Strengthening these semantic foundations could enhance traceability, reduce generic phrasing, and improve the assistant's ability to enforce level-appropriate detail.

Finally, evaluating end-to-end authoring, review, and approval cycles with practitioners inside a common data environment will help verify not only textual quality but also operational fit, creating the conditions for more conclusive statements in subsequent stages of the research. For example, it would be valuable to compare the capacity of the strategies described and develop through this thesis with other AI refinement strategies like Cache-Augmented Generation (CAG) and Fine-Tuning.

This page is intentionally left blank

REFERENCES

Journal article

Abualdenien, J. and Borrmann, A. (2022) 'Levels of detail, development, definition, and information need: a critical literature review'. Journal of Information Technology in Construction (ITcon), 27, pp. 363–392. doi: 10.36680/j.itcon.2022.018.

Katranuschkov, P., Gehre, A. and Scherer, R.J. (2003) 'An ontology framework to access IFC model data'. Journal of Information Technology in Construction (ITcon), 8, pp. 413–437.

Lee, Y.-C., Eastman, C.M. and Solihin, W. (2016) 'An ontology-based approach for developing data exchange requirements and model views of building information modeling'. Advanced Engineering Informatics, 30(3), pp. 354–367. doi: 10.1016/j.aei.2016.04.008.

Oliveira, A., Granja, J., Bolpagni, M., Motamedi, A. and Azenha, M. (2024) 'Development of standard-based information requirements for the facility management of a canteen'. Journal of Information Technology in Construction (ITcon), 29, pp. 281–307. doi: 10.36680/j.itcon.2024.014.

Du, S., Hou, L., Zhang, G., Tan, Y. and Mao, P. (2024) 'BIM and IFC data readiness for AI integration in the construction industry: a review approach'. Buildings, 14, 3305. doi: 10.3390/buildings14103305.

Mitera-Kiełbasa, E. and Zima, K. (2024) 'Automated classification of exchange information requirements for construction projects using Word2Vec and SVM'. Infrastructures, 9, 194. doi: 10.3390/infrastructures9110194.

Funk, M., Hosemann, S., Jung, J. C., and Lutz, C. (2023) 'Towards ontology construction with language models. *arXiv*, 2309.09898. Available at: https://arxiv.org/abs/2309.09898.

Höltgen, L., Zentgraf, S., Hagedorn, P., and König, M. (2025) 'Utilizing large language models for semantic enrichment of infrastructure condition data: a comparative study of GPT and Llama models. *AI in Civil Engineering*, 4:14. doi: 10.1007/s43503-025-00055-9.

Hou, X., Zhao, Y., Wang, S., and Wang, H. (2025) 'Model Context Protocol (MCP): Landscape, security threats, and future research directions' *arXiv*, 2503.23278. Available at: https://arxiv.org/abs/2503.23278.

Book

Eastman, C., Teicholz, P., Sacks, R., and Liston, K. (2008) 'BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers, and Contractors.' Hoboken, NJ: John Wiley & Sons.

Conference proceedings

Schilling, S and Clemen, C (2024) 'Standard-Oriented Ontology Export of Domain Catalogues from Data Dictionaries', in LDAC 2024: 12th Linked Data in Architecture and Construction Workshop, CEUR Workshop Proceedings, Bochum, Germany, 13–14 June. ISSN 1613-0073.

Mellenthin Filardo, M., Liu, L., Hagedorn, P., Zentgraf, S., Melzner, J., and König, M (2024) 'A Standard-Based Ontology Network for Information Requirements in Digital Construction Projects', in LDAC 2024: 12th Linked Data in Architecture and Construction Workshop, CEUR Workshop Proceedings, Bochum, Germany, 13–14 June. ISSN 1613-0073.

Poveda-Villalón, M., Carulli-Pérez, S. and García-Castro, R. (2024) 'How Much OWL Do You Need to Know to Make Sense of Building Ontologies', in LDAC 2024: 12th Linked Data in Architecture and Construction Workshop, CEUR Workshop Proceedings, Bochum, Germany, 13–14 June. ISSN 1613-0073.

Oraskari, J., Kirner, L., Zöcklein, M. and Brell-Cokcan, S. (2024) 'A Method to Unify Custom Properties in IFC to Linked Building Data Conversion', in LDAC 2024: 12th Linked Data in Architecture and Construction Workshop, CEUR Workshop Proceedings, Bochum, Germany, 13–14 June. ISSN 1613-0073.

Schulz, O., Oraskari, J. and Beetz, J. (2023) 'Lessons Learned from Designing and Using bcfOWL', in LDAC 2023: 11th Linked Data in Architecture and Construction, CEUR Workshop Proceedings, Matera, Italy, 15–16 June. ISSN 1613-0073.

Alexiev, V., Radkov, M. and Keberle, N. (2023) 'Semantic bSDD: Improving the GraphQL, JSON and RDF Representations of buildingSmart Data Dictionary', in LDAC 2023: 11th Linked Data in Architecture and Construction, CEUR Workshop Proceedings, Matera, Italy, 15–16 June. ISSN 1613-0073.

Teclaw, W., Rasmussen, M.H., Labonnote, N., Oraskari, J. and Hjelseth, E. (2023) 'The semantic link between domain-based BIM models', CEUR Workshop Proceedings (LDAC), ISSN 1613-0073.

Oraskari, J. (2021) 'Live Web Ontology for buildingSMART Data Dictionary', in 32. Forum Bauinformatik 2021, vol. 32, pp. 166–173. Online, 9–10 September. doi: 10.26083/tuprints-00019496.

Schulz, O., Oraskari, J. and Beetz, J. (2021) 'bcfOWL: A BIM collaboration ontology', in Proceedings of the 9th Linked Data in Architecture and Construction Workshop (LDAC 2021), CEUR Workshop Proceedings, Luxembourg, pp. 1–12. ISSN 1613-0073.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020) 'Retrieval-augmented generation for knowledge-intensive NLP tasks', in *Advances in Neural Information Processing Systems 34 (NeurIPS 2020)*, Vancouver, Canada.

Tomczak, A., van Berlo, L., Krijnen, T., Borrmann, A. and Bolpagni, M. (2022) 'A review of methods to specify information requirements in digital construction projects', in IOP Conference Series: Earth and Environmental Science, 1101, 092024. doi: 10.1088/1755-1315/1101/9/092024.

Xu, Z., Cruz, M. J., Guevara, M., Wang, T., Deshpande, M., Wang, X., and Li, Z. (2024) 'Retrieval-augmented generation with knowledge graphs for customer service question answering', in Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), Washington, DC, USA. doi: 10.1145/3626772.3661370.

Dissertations

Šimánek, P. (2024) 'IDS in Open BIM Workflow: Potential & Limitations'. Master's thesis, University of Minho, School of Engineering (European Master in Building Information Modelling – BIM A+), Portugal, September.

Standards

International Organization for Standardization. (2018). 'ISO 19650-1:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 1: Concepts and principles.' Geneva: ISO.

International Organization for Standardization. (2018). 'ISO 19650-2:2018 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 2: Delivery phase of the assets.' Geneva: ISO.

International Organization for Standardization. (2020). 'ISO 19650-3:2020 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 3: Operational phase of the assets.' Geneva: ISO.

International Organization for Standardization. (2022). 'ISO 19650-4:2022 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 4: Information exchange.' Geneva: ISO.

International Organization for Standardization. (2020). 'ISO 19650-5:2020 Organization and digitization of information about buildings and civil engineering works, including building information modelling (BIM) — Information management using building information modelling — Part 5: Security-minded approach to information management.' Geneva: ISO.

International Organization for Standardization. (2024). 'ISO 7817-1:2024 Building Information Modelling — Level of Information Need — Part 1: Concepts and principles.' Geneva: ISO.

International Organization for Standardization. (2016). 'ISO 29481-1:2016 Building information models — Information delivery manual — Part 1: Methodology and format.' Geneva: ISO.

International Organization for Standardization. (2020). 'ISO 23387:2020 Building information modelling (BIM) — Data templates for construction objects used in the life cycle of any built asset — Concepts and principles.' Geneva: ISO.

Website

buildingSMART International (n.d.) 'buildingSMART Data Dictionary (bSDD)' [Online]. Available at: https://www.buildingsmart.org/users/services/buildingsmart-data-dictionary/ (Accessed: 30 August 2025).

APPENDIX 1: DEVELOPMENT ROADMAP

Version	Date	Core Functionality & Approach	Main Limitations / Bugs	Deactivated / Deprecated Features
0.0.1	2025-04-08	Initial RAG pipeline with local Ollama LLM, PDF + ontology text ingestion	Hard-coded prompts; no conversational UI	None
0.0.2	2025-04-24	Added SPARQL-driven ontology class/label extraction; refined PDF chunking	Limited field classification; static QA chain	Direct ontology text parsing (moved to label-based parsing)
0.0.3	2025-04-24	Introduced organic conversational loop for LOIN field collection	Occasional misclassification; no user hints	Predefined prompt scaffolds (replaced by dynamic prompts)
0.0.4	2025-04-24	Improved question-generation logic; dynamic hints; fuzzy matching removed	User needed to rephrase often; limited ontology validation	Rapidfuzz checks (replaced)
0.0.5	2025-05-06	Full Q&A form flow; ISO-only prompt structure; replaced fuzzy classification	Manual Spanish prompts; missing field for documentation	Classification via tokenizer (replaced by simple loop)
0.0.6	2025-05-06	Streamlined ontology label extraction; simplified PDF ingestion	Single-step field input; no skipping; limited UX	Multi-step NLP checks
0.0.7	2025-05-07	Switched to OpenAI GPT-3.5-turbo; environmental variable for API key; improved UX	No validation against ontology; all fields forced	Ollama embeddings & LLM
0.0.8	2025-05-07	Added get_close_matches validation; field hints; skip option for fields	Occasional hint mismatch; limited ontology alignment	None
0.0.9	2025-05-07	Robust field-hints map; better skip handling; vectorstore build at runtime	No streaming UI; flat script, slow cold start	Interactive UI stubs

0.1.0	2025-05-07	Ontology graph TTL load; integrated ISO-PDF contextual QA chain	Non-structured prompts; markdown wrapper bug	Early form-like prompts
0.1.1	2025-05-07	(minor patch) lemmatizer+RapidFuzz synonym expansion; basic ontology label preprocessing	Incomplete synonym coverage; NLTK resource dependency errors	Hard-coded examples
0.1.2	2025-05-07	Field vocabularies added; interactive input validation & confirmation prompts	Valid_vocab too restrictive; user fatigue	Blind RAG skippable
0.1.3	2025-05-22	BSDD term fetch; dynamic vocabulary extension via API; ontology property checks	API rate limits; extended cold start times	Manual vocabulary lists
0.1.4	2025-05-22	SHACL placeholder; WebResource suggestions; suggestions on missing ontology classes	Unfinished SHACL; web scraping flaky	Fallback HTML scraping
0.1.5	2025-05-22	Node-based export mode; batch documentation generation; multi-question script	No conversational mode; limited to ontology	Single-question CLI
0.1.6	2025-05-22	SPARQL + local RAG QA chain unified; streamlined code structure	No user prompts; static Q&A	Legacy conversational loops
0.1.7	2025-05-22	Combined memory conversational RAG via Streamlit; summary + buffer memory	Requires Streamlit; heavier dependencies	CLI mode
0.1.8	2025-05-22	Modular conversational assistant; refined missing-field logic	Occasional miscategorization; no UI interface	Shallow BBDD validation
0.1.9	2025-05-22	DeepResearch CLI: DuckDuckGo web +	Slow; rate-limited; complex prompt assembly	Pure PDF-only mode

		PDF hybrid search; trust-domain scraping		
0.2.0	2025-05-08	ConversationalRetrievalChain; multi-PDF context; memory integration	No LOIN form flow; generic ISO bot	Direct LOIN table generation
0.2.1	2025-05-29	Streamlit UI; summary & buffer memory; live chat with ISO RAG	Web-app only; no offline mode; no LOIN-specific fields	CLI / script mode
0.2.2	2025-06-25	Deep Research with DuckDuckGo + caching; gpt-4o; trusted-domain web augmentation	External search delays; complex error handling	Pure local PDF context
1.1.1	2025-06-03	Full rewrite to node-based agent architecture: modular nodes for ingestion, RAG, convo	Initial node-orchestration bugs; missing ontology agent	Monolithic scripts
2.1.1	2025-06-15	Modular node-based pipeline; central orchestrator; LOIN form flow with prompt-based row generation; local RAG on preloaded standards; markdown chunking and embedding with vector DB; export to CSV.	No UI; limited prompt traceability; SHACL logic not implemented; form flow depends on strict template; weak error handling and memory persistence.	Web search present but unused; CLI test modules deprecated; chat-style prompting replaced by form node calls.
2.1.2	2025-06-17	Hybrid RAG pipeline with PDF context and web scraping; deep multi-level search from trusted sources; streaming GPT-4o output; markdown chunking and embedding with vector DB; modular test scripts for local and web queries.	No structured LOIN form or export; no SHACL or ontology validation; brittle link scraping; markdown structure can mislabel sections; search depth increases latency; lacks session persistence across runs.	LOIN form flow removed; CSV export logic removed; freeform local testing retained but isolated.
2.1.3	2025-06-25	Adds LOIN field parser with structured	No SHACL or ontology validation;	Deep search pipeline removed; legacy

		output; CSV export via template; prompt fallback to web if local context is weak; improved response formatting; modular RAG pipeline and markdown embedding retained.	missing field logic is LLM-dependent; CSV limited to one-row overwrite; no multi-entry mode; no UI; no live form editing; session history not persistent.	streaming loop removed; CLI tests still present but unused in main.
2.1.4	2025-07-01	Adds logging of full sessions; reintroduces deep web search with iterative refinement; modular RAG flow with summarization; clean prompt generation; local + web fallback logic integrated.	No SHACL/ontology checks; LOIN export still one-row only; no form UI; summary quality varies by source; no structured field validation; session logs are raw text.	None; previous removals (e.g. chat UI, XLSX export) remain out.
2.1.6	2025-07-02	Refined deep search with LLM-assisted fallback; strict domain filtering; MMR-based PDF retrieval; selective summary filtering; improved chunking with metadata; robust error handling and logging.	No ontology or SHACL checks; no UI; LOIN still limited to 1-row CSV; no XLSX or JSON-LD; summaries not integrated into LOIN export.	None; all modules refactored and in active use.
2.1.7	2025-07-03	Adds chat-driven LOIN extractor with CSV export; refined deep search with LLM-based fallback and deduplication; improved logging and prompt layering; web and PDF context merged into full assistant response.	No ontology checks; only 1-row LOIN export; no live form editing or XLSX; summaries not linked to export; no multi-turn memory; trusted source filter still static.	None; all modules in active use with clean CLI loop.
2.1.8	2025-07-04	Enables multi-entry CSV export from conversational LOIN extraction; improved	No ontology or SHACL checks; no UI; XLSX and JSON-LD not supported;	None; all modules are active and integrated.

		deep search with loop control and query rephrasing; chunk metadata added; prompt context fully layered; PDF+web fusion retained.	summaries not linked to table rows; sessions are ephemeral; BSDD integration missing.	
2.1.9	2025-07-06	Improved web and PDF summarization with filtered MMR-based retrieval; conversational LOIN flow backed by CSV schema; enhanced prompt composition from layered sources; structured logging and CLI loop retained.	No ontology validation or structured form UI; LOIN entries not linked to retrieved context; limited export options; memory non-persistent beyond session.	None; all modules active and integrated.
2.2.1	2025-07-08	Adds multi-stage pipeline: local answer, web query refinement, query decomposition, and summary aggregation; layered prompt composition; web summaries filtered; improved chunk indexing with metadata.	No ontology validation or UI; context not linked to LOIN fields; only CSV output; session memory not persistent across runs; BSDD integration absent.	None; older deep search logic restructured but retained as fallback mechanism.
2.2.2	2025-07-09	Adds. env config for full path/model control; expands trusted web sources; prevents redundant PDF reprocessing; improved metadata tagging in chunks; clearer prompt structure for GPT-4o with labeled context.	Still no ontology validation or form UI; no XLSX/JSON-LD output; BSDD integration absent; prompt logic depends on clean inputs; context not linked to LOIN rows.	None; internal structure cleaned, all functions retained.
2.2.3	2025-07-12	Fully modular site-specific search with query decomposition fallback; labeled	No SHACL/ontology logic; UI and export formats (XLSX/JSON-LD) missing; LOIN	None; all features from prior version retained and expanded.

		prompt sections; clean logging of search and response history; PDF chunks indexed with source+chunk metadata; all stages logged.	output not traceable to source context; BSDD not integrated.	
2.2.4	2025-07-20	Modular pipeline with trusted web domains and fallback control; refined prompt assembly with history/context layers; improved Markdown chunking with metadata; centralized. env configuration; complete search logging; concise summary filtering.	No ontology checks, UI, or export formats beyond CSV; source traceability and BSDD mapping missing; LOIN table not linked to retrieved evidence.	None; all modules active and used in the flow.
2.2.4.3	2025-08-10	Adds GUI via Streamlit interface for real-time LOIN table generation; integrates deepsearch and deepsearch_online for recursive query refinement (local and web); conversational LOIN flow now unified with PDF+web context; canonical ISO 7817 field order extracted to schema module; clean session flow combining agent logic, logging, and RAG outputs.	No ontology validation or semantic alignment to source; lacks export formats beyond CSV; GUI lacks field editing or preview; BSDD, JSON-LD, and SHACL still not implemented; traceability to chunk or source not visualized.	None; all core modules retained, modularized, and reused under new structure.

APPENDIX 2: PRELIMINARY EVALUATION TABLES

MODEL	AVG-RUN 1	AVG-RUN 2	AVG-RUN 3	AVG
DeepSeek-r1:latest	2.42	2.33	2.17	**2.31**
ChatGPT 5 Auto	2.83	2.67	3.08	**2.86**
ChatGPT 5 - Custom	3.83	3.42	3.25	**3.50**
M04	1.08	1.00	1.00	**1.03**
M05	0.00	0.00	0.00	**0.00**
M06	0.00	0.00	0.00	**0.00**
M07	3.42	3.08	3.08	**3.19**
M08	3.50	3.42	3.42	**3.44**
M09	3.75	3.83	3.83	**3.81**
M10	3.58	3.58	3.58	**3.58**
M11-9	1.00	1.00	1.00	**1.00**
M11-13	1.00	1.00	1.00	**1.00**
M11-24	1.00	1.00	1.00	**1.00**

Table 47 – Average Mean Score of Models in Test 1

MODEL	AVG-RUN 1	AVG-RUN 2	AVG-RUN 3	AVG
DeepSeek-r1:latest	1.76	1.81	1.78	**1.78**
ChatGPT 5 Auto	1.50	1.37	1.39	**1.42**
ChatGPT 5 - Custom	3.31	3.59	3.56	**3.49**
M4	1.15	1.09	1.04	**1.09**
M5	0.00	0.00	0.00	**0.00**
M6	0.11	0.11	0.11	**0.11**
M7	2.81	2.52	2.59	**2.64**
M8	3.83	3.93	4.15	**3.97**
M9	4.70	4.69	4.59	**4.66**
M10	4.78	4.78	4.76	**4.77**
M11-9	4.04	3.69	3.65	**3.79**
M11-13	3.69	3.89	3.69	**3.75**
M11-24	4.37	3.63	3.74	**3.91**

Table 48 – Average Mean Score of Models in Test 2

TEST 1 - AVG PARAM SCORE M1/M2/M3						
MOD	**A**	**B**	**C**	**D**	**E**	**F**
DeepSeek-r1:latest	2.17	2.50	2.83	2.00	2.00	2.33
ChatGPT 5 Auto	3.17	3.17	2.67	2.67	2.67	2.83
ChatGPT 5 - Dedicated GPT	4.00	3.00	3.83	3.17	4.00	3.00
TEST 2 - AVG PARAM SCORE M1/M2/M3						
MOD	**A**	**B**	**C**	**D**	**E**	**F**

DeepSeek-r1:latest	1.96	1.85	2.07	1.78	1.00	2.04
ChatGPT 5 Auto	1.37	1.59	1.56	1.26	1.00	1.74
ChatGPT 5 - Dedicated GPT	3.74	3.04	3.59	3.44	3.48	3.63
AVERAGE PARAMETER SCORE - M1/M2/M3						
MOD	**A**	**B**	**C**	**D**	**E**	**F**
DeepSeek-r1:latest	2.06	2.18	2.45	1.89	1.50	2.19
ChatGPT 5 Auto	2.27	2.38	2.11	1.96	1.83	2.29
ChatGPT 5 - Dedicated GPT	3.87	3.02	3.71	3.31	3.74	3.31

Table 49 – Average Parameter Mean Score M01-M02-M03

TEST 1 - AVG PARAM SCORE - M9/M10/M11						
MOD	A	B	C	D	E	F
M9	4.17	3.83	4.00	3.83	3.00	4.00
M10	4.00	4.00	3.50	3.50	3.00	3.50
M11-24	1.00	1.00	1.00	1.00	1.00	1.00
TEST 2 - AVG PARAM SCORE - M9/M10/M11						
MOD	A	B	C	D	E	F
M9	4.89	4.89	4.59	4.63	4.48	4.48
M10	4.89	4.89	4.85	5.00	4.33	4.67
M11-24	4.26	4.26	3.67	4.11	3.00	4.19
AVG PARAMTER SCORE - M9/M10/M24						
MOD	A	B	C	D	E	F
M9	4.53	4.36	4.30	4.23	3.74	4.24
M10	4.44	4.44	4.18	4.25	3.67	4.08
M11-24	2.63	2.63	2.33	2.56	2.00	2.59

Table 50 - Average Parameter Mean Score M09-M10-M11.24

AVERAGE PARAMETER MEAN SCORE - M11 CONFIGURATIONS								
CON	A	B	C	D	E	F	**AVG**	**AVP**
C01	4.00	4.00	3.11	3.70	3.00	4.00	**3.64**	**73%**
C02	3.74	3.59	3.81	3.48	2.74	3.70	**3.51**	**70%**
C03	3.63	3.67	3.22	3.26	2.93	3.74	**3.41**	**68%**
C04	3.56	3.67	3.22	3.22	3.00	3.70	**3.40**	**68%**
C05	3.48	3.52	3.44	3.37	3.00	3.78	**3.43**	**69%**
C06	3.67	3.67	3.19	3.19	3.00	3.70	**3.40**	**68%**
C07	4.07	4.07	3.81	3.48	3.00	3.89	**3.72**	**74%**
C08	3.81	3.81	3.56	3.22	3.00	3.52	**3.49**	**70%**
C09	4.15	4.15	3.56	3.85	3.00	4.04	**3.79**	**76%**
C10	3.52	3.52	3.26	3.37	3.00	3.67	**3.39**	**68%**
C11	3.78	3.89	3.63	3.70	3.04	3.78	**3.64**	**73%**
C12	3.85	4.00	3.37	3.70	3.00	4.04	**3.66**	**73%**
C13	4.07	4.07	3.41	3.96	3.00	4.00	**3.75**	**75%**
C14	3.85	3.85	3.48	3.81	3.00	4.00	**3.67**	**73%**

C15	3.22	3.56	3.33	3.19	3.00	3.19	**3.25**	**65%**
C16	3.74	3.85	3.11	3.70	3.00	3.70	**3.52**	**70%**
C17	4.00	4.00	3.22	3.78	3.00	4.00	**3.67**	**73%**
C18	4.04	4.04	3.37	3.74	3.00	4.00	**3.70**	**74%**
C19	4.04	4.04	3.26	3.81	3.00	4.00	**3.69**	**74%**
C20	4.00	4.00	3.15	3.70	3.00	4.00	**3.64**	**73%**
C21	3.93	3.93	3.33	3.74	3.00	3.89	**3.64**	**73%**
C22	4.00	4.00	3.30	3.89	3.00	4.00	**3.70**	**74%**
C23	4.00	4.00	3.11	3.78	3.00	4.00	**3.65**	**73%**
C24	4.26	4.26	3.67	4.11	3.00	4.19	**3.91**	**78%**
C25	3.81	3.93	3.33	3.67	3.00	3.70	**3.57**	**71%**
C26	4.00	4.00	3.11	3.78	3.00	4.00	**3.65**	**73%**
C27	4.00	4.00	3.19	3.89	3.00	4.00	**3.68**	**74%**

Table 51 – Average Mean Scores of M11 by Configuration.

STANDARD DEVIATION BY CONFIGURATION OF M11								
CON	**AD-A**	**AD-B**	**AD-C**	**AD-D**	**AD-E**	**AD-F**	**AD-AVG**	**AVG-DP**
C01	0.00	0.00	0.00	0.06	0.00	0.00	**0.01**	**0.2%**
C02	0.17	0.17	0.06	0.17	0.13	0.06	**0.06**	**1.2%**
C03	0.06	0.00	0.11	0.06	0.06	0.06	**0.05**	**1.0%**
C04	0.00	0.00	0.00	0.00	0.00	0.06	**0.01**	**0.2%**
C05	0.32	0.06	0.19	0.13	0.00	0.19	**0.02**	**0.4%**
C06	0.00	0.00	0.13	0.06	0.00	0.06	**0.02**	**0.4%**
C07	0.32	0.32	0.06	0.45	0.00	0.33	**0.24**	**4.8%**
C08	0.13	0.13	0.38	0.00	0.00	0.06	**0.12**	**2.4%**
C09	0.36	0.36	0.33	0.28	0.00	0.06	**0.21**	**4.3%**
C10	0.51	0.51	0.26	0.45	0.00	0.38	**0.35**	**7.1%**
C11	0.48	0.48	0.50	0.42	0.06	0.48	**0.40**	**7.9%**
C12	0.26	0.00	0.23	0.23	0.00	0.06	**0.07**	**1.3%**
C13	0.13	0.13	0.32	0.13	0.00	0.00	**0.12**	**2.4%**
C14	0.13	0.13	0.32	0.06	0.00	0.00	**0.00**	**0.0%**
C15	0.00	0.00	0.19	0.06	0.00	0.06	**0.05**	**1.1%**
C16	0.45	0.26	0.00	0.51	0.00	0.51	**0.29**	**5.8%**
C17	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.0%**
C18	0.06	0.06	0.26	0.26	0.00	0.00	**0.02**	**0.4%**
C19	0.06	0.06	0.06	0.06	0.00	0.00	**0.04**	**0.9%**
C20	0.00	0.00	0.06	0.06	0.00	0.00	**0.02**	**0.4%**
C21	0.32	0.32	0.19	0.45	0.00	0.19	**0.18**	**3.6%**
C22	0.00	0.00	0.13	0.00	0.00	0.00	**0.02**	**0.4%**
C23	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.0%**
C24	0.45	0.45	0.80	0.40	0.00	0.32	**0.40**	**8.0%**
C25	0.53	0.34	0.58	0.62	0.00	0.51	**0.40**	**8.0%**

C26	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.0%**
C27	0.00	0.00	0.13	0.00	0.00	0.00	**0.02**	**0.4%**

Table 52 - Standard Deviation of M11 by Configuration.

CONFIGURATIONS OF M11						
CON	**D**	**R**	**C**	**O**	**W**	**WD**
C01	0	0	0	0	0	0
C02	0	0	0	0	1	0
C03	0	0	0	0	1	1
C04	0	0	1	0	0	0
C05	0	0	1	0	1	0
C06	0	0	1	0	1	1
C07	0	0	1	1	1	0
C08	0	0	1	1	1	0
C09	0	0	1	1	1	1
C10	1	0	0	0	0	0
C11	1	0	0	0	1	0
C12	1	0	0	0	1	1
C13	1	0	1	0	0	0
C14	1	0	1	0	1	0
C15	1	0	1	0	1	1
C16	1	0	1	1	0	0
C17	1	0	1	1	1	0
C18	1	1	0	0	0	0
C19	1	1	0	0	1	0
C20	1	1	0	0	1	0
C21	1	1	0	0	1	1
C22	1	1	1	0	0	0
C23	1	1	1	0	1	1
C24	1	1	1	1	0	0
C25	1	1	1	1	1	0
C26	1	1	1	1	1	0
C27	1	1	1	1	1	1

Table 53 - List of Configurations of M11

AVERAGE SCORE AND STANDARD DEVIATION OF M3						
PROMPT	**RUN 1**	**RUN 2**	**RUN 3**	**AVG**	**SD**	**CV**
P1	3.33	3.83	3.83	**3.67**	**0.29**	**6%**
P2	3.17	3.67	3.67	**3.50**	**0.29**	**6%**
P3	3.17	3.83	3.50	**3.50**	**0.33**	**7%**
P4	3.00	3.17	3.17	**3.11**	**0.10**	**2%**
P5	3.17	3.50	4.00	**3.56**	**0.42**	**8%**
P6	3.67	3.83	3.67	**3.72**	**0.10**	**2%**

P7	3.67	3.50	3.33	**3.50**	**0.17**	**3%**
P8	3.33	3.50	3.50	**3.44**	**0.10**	**2%**
P9	3.33	3.50	3.33	**3.39**	**0.10**	**2%**

Table 54 - Average Score and Deviation of M3 by Prompt

AVERAGE SCORES AND STANDARD DEVIATION OF M10							
PROMPT	**RUN 1**	**RUN 2**	**RUN 3**	**AVG**	**SD**	**CV**	**CI**
P1	4.75	4.75	4.83	**4.78**	**0.05**	**1%**	**30.3%**
P2	4.75	4.75	4.83	**4.78**	**0.05**	**1%**	**36.5%**
P3	4.50	4.50	4.50	**4.50**	**0.00**	**0%**	**28.6%**
P4	4.75	4.75	4.83	**4.78**	**0.05**	**1%**	**53.6%**
P5	4.50	4.50	4.67	**4.56**	**0.10**	**2%**	**28.1%**
P6	4.25	4.25	4.17	**4.22**	**0.05**	**1%**	**13.4%**
P7	5.00	5.00	5.00	**5.00**	**0.00**	**0%**	**42.9%**
P8	5.00	5.00	5.00	**5.00**	**0.00**	**0%**	**45.2%**
P9	5.00	5.00	5.00	**5.00**	**0.00**	**0%**	**47.5%**

Table 55 - Average Scores and Deviation of M10 by Prompt, Including CI

AVERAGE SCORE AND STANDARD DEVIATION OF M3 BY PARAMETER						
PARAM	**RUN 1**	**RUN 2**	**RUN 3**	**AVG**	**SD**	**CV**
A	3.33	4.00	3.89	**3.74**	**0.36**	**7%**
B	2.89	3.22	3.00	**3.04**	**0.17**	**3%**
C	3.22	3.67	3.89	**3.59**	**0.34**	**7%**
D	3.44	3.56	3.33	**3.44**	**0.11**	**2%**
E	3.56	3.33	3.56	**3.48**	**0.13**	**3%**
F	3.44	3.78	3.67	**3.63**	**0.17**	**3%**

Table 56 - Average Score and Deviation of M3 by Evaluation Parameter

AVERAGE SCORES AND STANDARD DEVIATION OF M10 BY PARAMETER							
PARAM	**RUN 1**	**RUN 2**	**RUN 3**	**AVG**	**SD**	**CV**	**IMP**
A	4.89	4.89	4.89	**4.89**	**0.00**	**0%**	**30.7%**
B	4.89	4.89	4.89	**4.89**	**0.00**	**0%**	**61.0%**
C	4.89	4.89	4.78	**4.85**	**0.06**	**1%**	**35.1%**
D	5.00	5.00	5.00	**5.00**	**0.00**	**0%**	**45.2%**
E	4.33	4.33	4.33	**4.33**	**0.00**	**0%**	**24.5%**
F	4.67	4.67	4.67	**4.67**	**0.00**	**0%**	**28.6%**

Table 57 - Average Scores and Deviation of M10 by Evaluation Parameter, Including CI

APPENDIX 3: M10 INTERACTION RECORD

QUESTION 1	I need to define the information for some elements of my project
RESPONSE	To define the information for elements of your project according to the Level of Information Need (LOIN) standards, you should consider the following steps as outlined in the ISO 19650 and EN 17412-1 standards: 1. **Purpose (Why):** Determine the purpose of the information. For example, is it for visualization, analysis, or construction? This helps in defining the level of detail required. 2. **Information Delivery Milestone (When):** Identify when the information is needed. This could be at different stages such as preliminary design, detailed design, or construction. 3. **Information Receiver (Who):** Specify who will receive the information. This could be a designer, contractor, or facility manager. 4. **Information Provider (Who):** Identify who is responsible for providing the information, such as an architect, surveyor, or engineer. 5. **Object (What):** Define the object or element for which information is required, such as a wall, door, or HVAC system. 6. **Breakdown Structure (What):** Establish a classification or breakdown structure for organizing the information. 7. **Geometrical Information (How):** Determine the geometrical aspects of the information, including: - Detail - Dimensionality - Location - Appearance - Parametric behavior 8. **Alphanumerical Information (How):** Specify the alphanumerical content, including: - Identification - Information content 9. **Documentation (How):** Identify the set of documents required to support the information, which may include drawings, specifications, or reports. To implement this, you can use a structured table format to document the prerequisites and specify the level of information need for each element. This approach ensures clarity and consistency across the project, facilitating effective information management and exchange.
QUESTION 2	I'm working on a house design, but I don't know what data I should include

RESPONSE	When working on a house design and determining the Level of Information Need (LOIN) according to ISO 19650 and EN 17412-1 standards, it's important to specify the necessary data for each element of your project. Here's a structured approach to help you determine what data to include: 1. **Purpose (Why):** Identify the purpose of the information for each element. For example, is it for visualization, structural analysis, asset management, or construction? 2. **Information Delivery Milestone (When):** Determine the stage at which the information is required, such as during preliminary design, detailed design, or construction. 3. **Information Receiver (Who):** Specify who will receive the information, such as architects, engineers, contractors, or facility managers. 4. **Information Provider (Who):** Identify who is responsible for providing the information, such as architects, surveyors, or engineers. 5. **Object (What):** Define the specific object or element for which information is required, such as walls, doors, windows, or HVAC systems. 6. **Breakdown Structure (What):** Establish a classification or breakdown structure for organizing the information, which helps in managing and accessing the data efficiently. 7. **Geometrical Information (How):** Determine the geometrical aspects of the information, including: - **Detail:** Level of detail required, such as simplified volumes for early design or detailed geometry for construction. - **Dimensionality:** 2D or 3D representation. - **Location:** Absolute or relative positioning within the project. - **Appearance:** Realistic textures and materials if needed for visualization. - **Parametric Behavior:** Whether parametric capabilities are needed. 8. **Alphanumerical Information (How):** Specify the alphanumerical content, including: - **Identification:** Unique identifiers for each element, like wall type or door type. - **Information Content:** Additional properties or specifications, such as material types or dimensions. 9. **Documentation (How):** Identify any supporting documents required, such as drawings, specifications, or reports. By organizing your information needs according to these categories, you can ensure that all relevant data is collected and communicated effectively throughout the project lifecycle. This structured approach will help facilitate better decision-making and coordination among project stakeholders.
QUESTION 3	We are currently developing the plans for the contractor
RESPONSE	When developing plans for the contractor in line with the ISO 19650 standards, it is important to ensure that the information management processes are clearly

	defined and organized. Here are some key considerations to help guide the development of these plans: 1. **BIM Execution Plan (BEP):** The contractor should prepare a BIM Execution Plan that outlines how information will be managed and delivered throughout the project. This includes: - The delivery team's approach to meeting the appointing party's exchange information requirements. - Objectives and goals for the collaborative production of information. - An overview of the delivery team's organizational structure and commercial relationships. 2. **Information Container Attributes:** Ensure that each information container has the necessary attributes assigned, such as: - Status (suitability) - Revision - Classification (according to ISO 12006-2) - Controlled access at the information container level 3. **Information Delivery Strategy:** The contractor should establish an information delivery strategy that details: - The timeline and milestones for information delivery. - The roles and responsibilities of each team member in the information management process. 4. **Security Considerations:** If applicable, develop a security-minded approach to protect sensitive information from unauthorized access or hostile reconnaissance, as outlined in ISO 19650-5. 5. **Common Data Environment (CDE):** Implement a Common Data Environment to facilitate secure information sharing among all project stakeholders. It is recommended that the CDE is in place before issuing invitations to tender. 6. **Communication and Coordination:** Establish clear communication channels and coordination mechanisms to ensure that all parties are aligned and informed about project developments and information requirements. By addressing these aspects, the contractor can effectively manage and deliver the required information in accordance with ISO 19650 standards, facilitating smooth project execution and collaboration among all stakeholders.
QUESTION 4	What does 'purpose' mean here? What should I choose?
RESPONSE	In the context of the Level of Information Need (LOIN) as outlined in ISO 19650 and EN 17412-1 standards, 'purpose' refers to the specific reason or intent for which information is required for a particular element or object in your project. It helps in determining the level of detail and type of information necessary to fulfill the project's requirements. The purpose can vary depending on the stage of the project and the needs of different stakeholders. When choosing a 'purpose,' consider the following:

1. **Functionality:** What is the information intended to achieve? This could be for design, analysis, construction, operation, maintenance, or visualization.

2. **Stakeholder Needs:** Different stakeholders may have different purposes for the information. For example, a designer might need detailed geometry for design purposes, while a contractor might need quantity information for cost estimation.

3. **Project Phase:** The purpose may change depending on the phase of the project, such as concept design, detailed design, or construction.

4. **Security Considerations:** In some cases, the purpose might not be explicitly disclosed to all actors for security reasons. It should be marked as "not disclosed" and only shared with authorized personnel.

5. **Organizational and Project Requirements:** The purposes can be derived from organizational information requirements or project-specific information requirements.

By clearly defining the purpose of the information, you can ensure that the right level of detail and type of information is provided, facilitating better decision-making and coordination among project stakeholders.

QUESTION 5	Okay, let's say it's for cost estimation
RESPONSE	When the purpose of the information is for cost estimation, you need to focus on gathering data that directly impacts the financial aspects of the project. According to the ISO 19650 and EN 17412-1 standards, here's how you can structure your Level of Information Need (LOIN) for cost estimation: 1. **Purpose (Why):** The primary purpose is cost estimation. This involves determining the financial requirements for the materials, labor, and other resources needed for the project. 2. **Information Delivery Milestone (When):** Identify the stage at which cost estimation is required. This could be during the preliminary design for budgetary purposes or during detailed design for more accurate cost forecasting. 3. **Information Receiver (Who):** The information will typically be received by cost estimators, quantity surveyors, or financial managers responsible for budgeting and financial planning. 4. **Information Provider (Who):** Architects, engineers, or contractors who have detailed knowledge of the design and construction requirements will provide the necessary information. 5. **Object (What):** Specify the elements or objects for which cost information is needed, such as walls, doors, windows, and structural components. 6. **Breakdown Structure (What):** Use a classification system to organize the cost information, such as a cost breakdown structure (CBS) or a work breakdown structure (WBS).

	7. **Geometrical Information (How):** Focus on the geometrical data that affects cost, such as dimensions and quantities. Detailed geometry might not be necessary unless it impacts the cost significantly. 8. **Alphanumerical Information (How):** Include: - **Identification:** Unique identifiers for each element to ensure accurate cost tracking. - **Information Content:** Material specifications, quantities, unit costs, labor rates, and any other financial data relevant to cost estimation. 9. **Documentation (How):** Provide supporting documents such as cost reports, material take-offs, and specifications that justify the cost estimates. By focusing on these aspects, you ensure that the information collected is sufficient to perform accurate cost estimations, helping stakeholders make informed financial decisions throughout the project lifecycle.
QUESTION 6	Which elements should I define information for?
RESPONSE	When defining information for elements in your project, especially when the purpose is cost estimation, you should focus on elements that significantly impact the financial aspects of the project. According to the ISO 19650 and EN 17412-1 standards, consider the following types of elements: 1. **Structural Elements:** - Walls, columns, beams, and floors - Foundation components - Roof structures 2. **Building Envelope:** - Exterior walls and insulation - Windows and doors - Roofing materials 3. **Interior Elements:** - Partitions and interior walls - Ceilings and flooring materials - Stairs and railings 4. **Mechanical, Electrical, and Plumbing (MEP) Systems:** - HVAC systems - Plumbing fixtures and piping - Electrical wiring and lighting fixtures 5. **Finishes and Fixtures:** - Surface finishes (e.g., paint, tiles) - Built-in furniture or cabinetry - Sanitary fixtures 6. **Site Elements:** - Landscaping and external works - Paving and parking areas - Drainage systems

For each of these elements, you should gather geometrical and alphanumerical information relevant to cost estimation, such as dimensions, quantities, material specifications, unit costs, and labor rates. Organize this information using a breakdown structure like a cost breakdown structure (CBS) or a work breakdown structure (WBS) to ensure clarity and facilitate accurate cost tracking and estimation.

QUESTION 7	Let’s go with walls, windows, and lights
RESPONSE	When defining the Level of Information Need (LOIN) for walls, windows, and lights, especially with a focus on cost estimation, you should consider the following aspects based on the ISO 19650 and EN 17412-1 standards: ### Walls 1. **Purpose (Why):** Cost estimation for materials and labor associated with wall construction. 2. **Information Delivery Milestone (When):** Required during detailed design for accurate cost forecasting. 3. **Information Receiver (Who):** Cost estimators or quantity surveyors. 4. **Information Provider (Who):** Architects or structural engineers. 5. **Object (What):** Walls, including exterior and interior partitions. 6. **Breakdown Structure (What):** Classification according to a cost breakdown structure (CBS) or work breakdown structure (WBS). 7. **Geometrical Information (How):** - **Detail:** Simplified volume representation including openings. - **Dimensionality:** 3D. - **Location:** Absolute positioning. - **Appearance:** Realistic with texture of materials. - **Parametric behavior:** Not requested. 8. **Alphanumerical Information (How):** - **Identification:** Wall type. - **Information Content:** Material specifications, dimensions, and quantities. 9. **Documentation (How):** Supporting documents such as material take-offs and specifications. ### Windows 1. **Purpose (Why):** Cost estimation for window installation and materials. 2. **Information Delivery Milestone (When):** Required during detailed design for cost forecasting. 3. **Information Receiver (Who):** Cost estimators or quantity surveyors.

4. **Information Provider (Who):** Architects or facade engineers.

5. **Object (What):** Windows, including frames and glazing.

6. **Breakdown Structure (What):** Classified according to CBS or WBS.

7. **Geometrical Information (How):**
   - **Detail:** Simplified box representation for asset management.
   - **Dimensionality:** 3D.
   - **Location:** Relative positioning in relation to walls.
   - **Appearance:** Realistic with transparency and reflectance.
   - **Parametric behavior:** Not requested.

8. **Alphanumerical Information (How):**
   - **Identification:** Window type.
   - **Information Content:** Frame material, glazing type, dimensions, and quantities.

9. **Documentation (How):** Material specifications and cost reports.

Lights

1. **Purpose (Why):** Cost estimation for lighting fixtures and installation.

2. **Information Delivery Milestone (When):** Required during detailed design for cost forecasting.

3. **Information Receiver (Who):** Cost estimators or quantity surveyors.

4. **Information Provider (Who):** Electrical engineers or lighting designers.

5. **Object (What):** Lighting fixtures and systems.

6. **Breakdown Structure (What):** Classified according to CBS or WBS.

7. **Geometrical Information (How):**
   - **Detail:** Simplified representation for cost estimation.
   - **Dimensionality:** 3D.
   - **Location:** Absolute positioning.
   - **Appearance:** Symbolic or realistic depending on visualization needs.
   - **Parametric Behavior:** Not requested.

8. **Alphanumerical Information (How):**
   - **Identification:** Light fixture type.
   - **Information Content:** Fixture specifications, quantities, and installation costs.

9. **Documentation (How):** Specifications, cost reports, and installation guidelines.

By defining the LOIN for these elements, you ensure that relevant and sufficient data is available for accurate cost estimation, helping stakeholders make informed financial decisions throughout the project lifecycle.

QUESTION 8	Can you give me a table with the information I need?
RESPONSE	Certainly! Based on the ISO 19650 and EN 17412-1 standards, I can provide you with a structured table format to define the Level of Information Need (LOIN) for walls, windows, and lights, focusing on cost estimation. Here's how you can organize the information: | **Element** | **Purpose (Why)** | **Information Delivery Milestone (When)** | **Information Receiver (Who)** | **Information Provider (Who)** | **Breakdown Structure (What)** | **Geometrical Information (How)** | **Alphanumerical Information (How)** | **Documentation (How)** | |---|---|---|---|---|---|---|---|---| | **Walls** | Cost estimation | Detailed design | Cost estimators, Quantity surveyors | Architects, Structural engineers | CBS/WBS | - Simplified volume representation - 3D - Absolute positioning - Realistic material texture - No parametric behavior | - Wall type ID - Material specifications - Dimensions and quantities | Material take-offs, Specifications | | **Windows** | Cost estimation | Detailed design | Cost estimators, Quantity surveyors | Architects, Facade engineers | CBS/WBS | - Simplified box representation - 3D - Relative positioning - Realistic appearance with transparency - No parametric behavior | - Window type ID - Frame material, glazing type - Dimensions and quantities | Material specifications, Cost reports | | **Lights** | Cost estimation | Detailed design | Cost estimators, Quantity surveyors | Electrical engineers, Lighting designers | CBS/WBS | - Simplified representation - 3D - Absolute positioning - Symbolic or realistic appearance | - Light fixture type ID - Fixture specifications - Quantities and installation costs | Specifications, Cost reports, Installation guidelines | This table provides a clear framework for documenting the necessary information for each element, ensuring that all relevant data is available for accurate cost estimation. You can use this format to populate the specific details for your project, ensuring consistency and clarity in information management.
QUESTION 9	Oh, also include doors
RESPONSE	Certainly! Here's an updated table including doors, along with walls, windows, and lights, focusing on cost estimation in line with ISO 19650 and EN 17412-1 standards: | **Element** | **Purpose (Why)** | **Information Delivery Milestone (When)** | **Information Receiver (Who)** | **Information Provider (Who)** | **Breakdown Structure (What)** | **Geometrical Information (How)** | **Alphanumerical Information (How)** | **Documentation (How)** | |---|---|---|---|---|---|---|---|---| | **Walls** | Cost estimation | Detailed design | Cost estimators, Quantity surveyors | Architects, Structural engineers | CBS/WBS | - Simplified volume representation - 3D - Absolute positioning - Realistic material texture - No parametric behavior | - Wall type ID - Material specifications - Dimensions and quantities | Material take-offs, Specifications |

Windows	Cost estimation	Detailed design	Cost estimators, Quantity surveyors	Architects, Facade engineers	CBS/WBS	- Simplified box representation - 3D - Relative positioning - Realistic appearance with transparency - No parametric behavior	- Window type ID - Frame material, glazing type - Dimensions and quantities	Material specifications, Cost reports
Lights	Cost estimation	Detailed design	Cost estimators, Quantity surveyors	Electrical engineers, Lighting designers	CBS/WBS	- Simplified representation - 3D - Absolute positioning - Symbolic or realistic appearance	- Light fixture type ID - Fixture specifications - Quantities and installation costs	Specifications, Cost reports, Installation guidelines
Doors	Cost estimation	Detailed design	Cost estimators, Quantity surveyors	Architects, Structural engineers	CBS/WBS	- Simplified box representation - 3D - Absolute positioning - Realistic appearance with material textures - No parametric behavior	- Door type ID - Material specifications - Dimensions and quantities	Material specifications, Cost reports

This table provides a structured approach to document the necessary information for each element, ensuring that all relevant data is available for accurate cost estimation. You can use this format to populate the specific details for your project, ensuring consistency and clarity in information management.