Enhancing Educational Support for JetBrains MPS with a Retrieval-Augmented LLM Chatbot: A Structured Knowledge Integration Approach

Sofia Meacham[1][a], Keith Phalp[2]
[1] *School of Computing & Engineering,Bournemouth University , Fern Barrow, Poole, UK*
[2] *Faculty of Science And Technology, Bournemouth University, Fern Barrow, Poole, UK*
smeacham@bournemouth.ac.uk, KPhalp@bournemouth.ac.uk

Keywords: Model-Based Software Engineering (MBSE), JetBrains MPS, Retrieval-Augmented Generation (RAG), LlamaIndex, Educational chatbots

Abstract: Model-Based Software Engineering (MBSE) with JetBrains MPS is challenging primarily because language engineering goes beyond using programming languages to designing them—working with meta-concepts, generators, and composition—so the learning curve is steep even with detailed documentation. We present an LLM-powered, retrieval-augmented chatbot for MPS education that combines official docs with expert-curated material, organizing both via composable graph indexes in LlamaIndex. We evaluate five configurations across two phases using the RAGAs framework along four dimensions: faithfulness, answer relevancy, context utilization, and harmfulness. Compared to a documentation-only baseline, faithfulness improves from 0.42 to 0.99; best context utilization reaches 0.71; answer relevancy remains 0.50–0.64 in the larger study; and harmfulness is as low as 0.05 (0.08 in the final configuration). These results indicate that (i) curated expert knowledge—beyond official docs—is crucial for onboarding to meta-level concepts, (ii) composable graphs materially improve grounding, and (iii) lightweight, targeted index summaries further boost reliability while remaining scalable. The approach generalizes to other MBSE tools where steep learning curves limit adoption, and we provide code and configuration artifacts to facilitate replication and classroom use.

1 INTRODUCTION

Model-Based Software Engineering (MBSE) with JetBrains MPS is challenging primarily because *language engineering* goes beyond using programming languages to *designing them*. Learners must reason about meta-concepts (e.g., abstract syntax, type systems), generators, and language composition, which makes the learning curve steep even when detailed documentation is available (Voelter and Solomatov, 2010; Barash and Pech, 2021; Prinz, 2021). In practice, newcomers benefit not only from official docs but also from *expert experience* that contextualizes why and how to apply MPS idioms (e.g., generator patterns, priorities, predefined plans) in real projects.

We present an LLM-powered, retrieval-augmented chatbot for MPS education that combines official documentation with expert-curated material and organizes both using *composable graph* indexes in LlamaIndex. This design addresses two key limitations of LLM-only assistants: (i) the need for curated, domain-specific grounding beyond generic tutorials (mps, 2024; Dummann and Silveira, 2024), and (ii) context-window constraints that hinder faithful use of larger corpora (Alarcia et al., 2024). By encoding brief, targeted *index summaries* and composing multiple indexes, our assistant retrieves and stitches together the most relevant fragments for a user's query.

We evaluate five configurations across two phases using the RAGAs framework (Es et al., 2024), reporting four dimensions—*faithfulness*, *answer relevancy*, *context utilization*, and *harmfulness*. Compared to a documentation-only baseline, faithfulness improves from 0.42 to 0.99; best context utilization reaches 0.71; answer relevancy remains 0.50–0.64 in the larger study; and harmfulness is as low as 0.05 (0.08 in the final configuration). These results indicate that (i) curated expert knowledge—beyond official docs—is crucial for onboarding to meta-level concepts, (ii) composable graphs materially improve grounding, and (iii) lightweight, targeted index sum-

[a] https://orcid.org/0000-0002-8474-4917

maries further boost reliability while remaining scalable.

Contributions. This paper offers three contributions to the MODELSWARD community:

1. A retrieval-augmented LLM assistant for JetBrains MPS that integrates official documentation and expert-curated sources via *composable graph* indexes in LlamaIndex (LlamaIndex Contributors, 2024b).
2. A metrics-based evaluation with *RAGAs* (Es et al., 2024) over two experiment phases, quantifying faithfulness, answer relevancy, context utilization, and harmfulness.
3. A practical recipe—source curation, targeted index summaries, and index composition—that instructors and practitioners can reuse to support MBSE tool adoption (e.g., generator maintainability, predefined plans) (Bucchiarone et al., 2021).

Please note that this paper focuses on the technical design and evaluation of the approach; a systematic study of its educational impact is deferred to future work.

The remainder of the paper is organized as follows: Section 2 summarizes background and research questions. Section 3 details the method and experimental setup. Section 4 reports results and discusses implications. Section 5 concludes and outlines future work.

2 BACKGROUND STUDY & RESEARCH QUESTIONS

Chatbots powered by Large Language Models (LLMs) are increasingly finding their place in Model-Based Software Engineering (MBSE) education, offering innovative ways to enhance learning and accessibility. MBSE, which emphasizes the use of models as primary artefacts in software development (Stahl et al., 2006), is a complex and technical field often associated with a steep learning curve, especially for beginners in academia and industry (Butting et al., 2018). The challenges of MBSE education stem from the need to grasp abstract concepts, specialized tools, and workflows that demand both theoretical and practical expertise. Chatbots provide an effective solution to these difficulties by acting as on-demand, interactive assistants that can simplify complex ideas, answer technical questions, and offer personalized learning experiences (Yigci et al., 2024; Micheal et al., 2024). By leveraging advanced capabilities such as Retrieval-Augmented Generation (RAG), these chatbots can deliver precise, context-aware information from diverse sources, helping students and professionals navigate MBSE's intricacies. As such, the integration of chatbots into MBSE education has the potential to transform how learners engage with this challenging subject, making it more approachable and engaging as Kienzle et al. mentioned in section 4.4.1 for Modelling Assistants (Kienzle et al., 2024).

Language engineering is a subfield of MBSE which is particularly challenging and highly specialised. Despite their potential, certain tools, such as MPS which is a language workbench developed by JetBrains (JetBrains, 2024), remain hard to learn and adopt both for academia and industry. MPS is powerful mainly for its language modularization and composition (Voelter and Solomatov, 2010), but complex tool for language engineering, widely recognized for its steep learning curve. Pech (Pech et al., 2013) clearly articulates MPS's potential and challenges in the closing statements of the paper, stating: *"Despite its relatively steep learning curve, MPS could help the adoption of the Language-Oriented Programming (LOP) principles among software practitioners and move the industry forward by making programming languages truly customizable."* The official JetBrains MPS documentation acknowledges this challenge, stating that "*MPS is quite different from what we've all known before so, in order to ease the learning curve of MPS for new users, we've created a Tutorial.*" in MPS JetBrains Blog (Toporov, 2008). This indicates that the developers recognize the difficulties users may face when starting with MPS and have provided resources to assist in the learning process. According to Barash and Pech, who authored an experience report on teaching MPS in both academic and industrial settings (Barash and Pech, 2021), learners face challenges such as a lack of foundational knowledge, the complexity of language engineering concepts, a steep learning curve, the need to adapt to unique techniques in MPS, and potential issues with motivation and engagement. These factors can significantly impact their ability to learn and effectively use MPS. Prinz in (Prinz, 2021) states explicitly in the experiences and evaluation section that MPS may work well for experienced developers but not for novice programmers. He also states that MPS is a heavy tool to use in teaching and the university students take a long time to get used to the tool. This difficulty is further exacerbated by well-known prerequisites within the community for learning MPS and language engineering — such as high-level modelling skills and advanced programming experience — which often lead to a high abandonment rate among both academic and

industrial users. Addressing these shortcomings is critical for ensuring broader adoption and effective use of tools like MPS.

To tackle these challenges, we focused on training an LLM to create a chatbot specifically designed to assist with MPS education. Our experiments began with the official MPS documentation (mps, 2024), which we subsequently enriched with a specialized blog addressing the complex topic of maintainable MPS generators (Dummann and Silveira, 2024). This advanced and essential feature of MPS is highly relevant to large-scale industrial projects, as demonstrated by the case studies and discussions presented in Part I, *JetBrains MPS in Industrial Applications*, of (Bucchiarone et al., 2021). Especially, complex variability-related systems as in (Schindler et al., 2021) consist of complex generators that should be maintained over time. Other systems that contain a large amount of DSLs and corresponding generators also require advanced MPS generator knowledge and the design and development of maintainable generators such as the set of DSLs in (Ratiu et al., 2021) by Ratiu et al. Barash and Pech (Barash and Pech, 2021) emphasize that mastering the generator is crucial for leveraging MPS's full potential in language development and code generation. While other online courses and topics were available, we opted to streamline our approach by concentrating on these two sources and their integration into the LLM. The proposed method is easily scalable and adaptable to other advanced MPS topics, such as language composition very well covered in a GitHub repository by Markus Voelter with a YouTube series (Voelter, 2024).

In the process of creating the chatbot, the inherent token limitations in LLMs posed a significant barrier to creating a robust and comprehensive chatbot. Studies have shown that token constraints can limit the depth and breadth of context that LLMs can process, impacting their ability to generate accurate and meaningful responses (Alarcia et al., 2024). Therefore, additional strategies were required to optimize the chatbot's performance.

The LlamaIndex library (LlamaIndex Contributors, 2024b) was employed to overcome these token limitations by adopting a retrieval-based approach rather than traditional prompting. LlamaIndex enables the integration of multiple sources of information without token constraints, facilitating a scalable and context-aware chatbot system. This approach allows for a seamless combination of documentation, blogs, and other resources, ensuring a more comprehensive and tailored learning experience for MPS users.

Through a series of experiments, we demonstrated that carefully designed composable graphs within LlamaIndex significantly enhance chatbot performance. These graphs facilitate the effective integration of domain-specific knowledge while maintaining high contextual relevance and accuracy. This method provides a scalable and efficient way to improve educational tools for highly technical domains, offering a promising solution for the limitations of traditional LLM approaches.

Building on this background, this research work explores the following research questions:

RQ1: How can retrieval-based techniques, such as those enabled by LlamaIndex, overcome token limitations in LLMs to enhance chatbot performance in MBSE education?

RQ2: What role do composable graphs play in improving the integration and retrieval of domain-specific knowledge for educational chatbots?

By addressing these questions, this work aims to contribute to the development of scalable and efficient educational tools that mitigate the challenges of MBSE education and foster broader adoption of tools like MPS.

3 METHOD

In this paper, we conducted a series of experiments and identified the highest-performing approach as our proposed method.

We based our work on the LlamaIndex library, which is provided as GitHub open source in (LlamaIndex Contributors, 2024a). LlamaIndex was created to address the token limitations often encountered with the LLMs context window limits when using prompting methods as was raised by Alarcia et al. in (Alarcia et al., 2024). It therefore allows the inclusion of numerous documents and other input sources without token constraints as part of context-augmentation stated in (LlamaIndex Contributors, 2024b). Despite this drawback, LlamaIndex provides significant benefits, including the ability to ingest and synthesize multiple sources. For source ingestion, we utilized LlamaHub (LlamaHub Contributors, 2024).

In subsection 3.1, we outline the high-level steps common to all experiments. Subsection 3.2 then provides a detailed description of the experimental design, followed by subsection 3.3, which details the experiments. In subsection 3.4, the evaluation method using the RAGAs framework is detailed.

3.1 Method steps - all experiments

Our experiments consisted of three main steps that were common to all and are detailed as follows:

Step 1: Ingestion of data

In this step, we ingest information in various formats supported by LlamaIndex and LlamaHub.

The structure and combination of one or more indexes at the data ingestion stage are critical for defining our experiments. Domain-specific information relevant to the problem is added during ingestion, an efficient approach compared to embedding domain-specific information at query time. Traditional methods apply transformations after data ingestion, which is resource-intensive and challenging to manage as was also claimed by Alekh Jindal et al. in (Jindal et al., 2017).

Step 2: Creation of a Query Engine

In this step, we build a query engine for the index or indexes created in the previous step and store it in a Pinecone vector database (Pinecone Team, 2024). Pinecone was selected for its ease of use and suitability for our documentation ingestion requirements. Pinecone is designed for ease of use, offering a user-friendly API and built-in security measures. However, it has some limitations regarding metadata handling and flexibility compared to other databases. In contrast, Neo4j utilizes the Cypher query language, which, while powerful for complex graph operations, can introduce additional complexity for users not requiring such advanced functionalities (DB-Engines, 2024).

Step 3: Evaluation Using the RAGAs Method

For evaluation, we utilized the RAGAs evaluation framework. A detailed explanation of the evaluation approach using the RAGAs framework is provided in Section 3.4.

3.2 Experiment Design

There were two main pieces of information used to train the LLM:

1. Official MPS documentation website (mps, 2024).
2. A blog dedicated to maintainable MPS generators (Dummann and Silveira, 2024).

The choice of these pieces of information originated from their importance in language engineering, the difficulty with the benefits of understanding MPS generators' mechanisms. MPS generator is the heart of the code-to-model connection where you can maintain your system model and the generated code in the same environment.

The official MPS documentation is a comprehensive documentation to refer to and contains information about the MPS generator too. The maintainable MPS generator part is an advanced topic and is not included in the main MPS documentation.

Likewise, this list of information sources can be extended to include more specialised topics. However, if the proposed approach is followed, the LLM will be able to be "trained" for all these topics appropriately. Taking it one step further, a chatbot that could complement the official documentation could be provided to students and industrial language engineers.

We conducted two sets of experiments. The first set consisted of three experiments each evaluated using four manually created questions and model-answers. The second set comprised five experiments, evaluated with a larger set of questions and model-answers, generated automatically through another LLM and manually assessed—so-called Human-in-the-loop generation as defined in (Wu et al., 2022) and used by many machine learning researchers.

In the first set, we used a smaller, representative set of questions created manually by the author. This allowed us to draw initial insights and conclusions about each approach.

The second set employed a dataset of question/model-answer pairs that was automatically created using a script. This script leveraged another LLM, gpt-4o, to generate the question/model-answer pairs, while all other operations were performed using the gpt-4 model. We ended up with 46 questions and we then manually corrected them following the Human-in-the-loop approach to enhance the credibility of our results. The human-in-the-loop validation was performed by a single domain expert with extensive experience in JetBrains MPS, language engineering, and generator development; the expert's role was limited to removing irrelevant questions and correcting factual inaccuracies. The corrected dataset consisted of a total of 39 questions. There were two types of corrections: one was to delete whole questions as irrelevant and the other was to modify the answer as incorrect. For example, we had to remove questions related to the webpage and not the concepts of the maintainable MPS generator such as: *"what is the purpose of the script included in the document, and how does it affect the display of the webpage?"* Also, we corrected one question whose automatically generated answer was very general and didn't actually answer the question.

All experiments are presented in the following subsections.

3.3 Experiments

The experiments follow an incremental design in which the first experiment of each set serves as a baseline. Subsequent experiments introduce targeted modifications to the ingestion phase and indexing strategy. Table 1 summarizes the key differences across experiments.

3.3.1 First set of experiments: small number, manually generated

The question/model-answer pairs used in this set were created manually by the first author based on extensive technical experience with the MPS generator. In this first set, Experiment 1_1 serves as the baseline, while Experiments 1_2 and 1_3 introduce targeted changes to the ingestion and indexing configuration (Table 1).

Experiment_1_1 (baseline). Only the official MPS documentation was ingested. A script was used to automatically download and parse all HTML pages of the documentation into a local folder, which was then indexed using LlamaIndex to create a single index.

Experiment_1_2 (added source, single merged index). Compared to Experiment 1_1, we additionally ingested the maintainable MPS generator blog and indexed both sources together into a single merged index.

Experiment_1_3 (multi-index with composable retrieval). Compared to Experiment 1_2, we ingested the two sources into separate indexes and combined them via a LlamaIndex `ComposableGraph`, enabling multi-index retrieval guided by index summaries.

3.3.2 Second set of Experiments: large number, semi-automatically generated

In all experiments, as a last step, we evaluated them using the manually corrected automatically generated questions that are described in the Experiment design.

The experiments in the second set follow an incremental design, where each configuration introduces a targeted modification over the baseline.

Experiment_2_1 (baseline). Only the official MPS documentation was ingested and indexed.

Experiment_2_2 (added source, single index). Compared to Experiment 2_1, the maintainable MPS generator blog was added and merged with the documentation into a single index.

Experiment_2_3 (separate indexes with composable graph). Compared to Experiment 2_2, the documentation and the generator blog were ingested into separate indexes and combined using a LlamaIndex `ComposableGraph`. Manually defined index summaries were introduced to guide query routing between the indexes; the corresponding composable structure is shown in the following listing.

```
composable_graph = ComposableGraph.
  from_indices(
    root_index_cls=
      GPTSimpleKeywordTableIndex,
    children_indices=[mpdsdoc_index,
      mpdsother_index],
    index_summaries=[
        "This is the main MPS documents
           for most questions.",
        "Use this for questions related
           to maintainable MPS
          generator, "
        "best practices for generator
          such as predefined generation
           plans. "
        "Also, use this for questions
          related to Mbeddr Example."
    ]
)
```

Experiment_2_4 (refined summaries). Compared to Experiment 2_3, the same composable graph configuration was retained, but the index summary for the maintainable MPS generator blog was refined to be more specific and keyword-oriented in order to improve query routing and retrieval quality. The updated summary is shown in the following listing.

```
composable_graph = ComposableGraph.
  from_indices(
    root_index_cls=
      GPTSimpleKeywordTableIndex,
    children_indices=[mpdsdoc_index,
      mpdsother_index],
    index_summaries=[
        "Use this for general questions
           about MPS.",
        "Use this for questions related
           to MPS generator.
          Maintainable, IF, LOOP, "
        "SWITCH, generator
          configuration, Mbeddr some of
           the keywords."
    ]
)
```

Experiment_2_5 (proposed approach). Compared to Experiment 2_4, the composable graph configuration was retained, but the index summaries were generated semi-automatically using an LLM and lightly adjusted to ensure coverage of key terms. This approach enables scalable summary creation for larger document collections and achieved the best overall performance. The resulting summaries are shown in the following listing.

Table 1: Summary of ingestion and indexing configurations across experiments.

Experiment	Data Sources	Indexing Strategy	Composable Graph	Index Summaries
1_1 / 2_1 (Baseline)	MPS documentation only	Single index	No	None
1_2 / 2_2	Documentation + generator blog	Single merged index	No	None
1_3 / 2_3	Documentation + generator blog	Separate indexes	Yes	Manual
2_4	Documentation + generator blog	Separate indexes	Yes	Keyword-oriented
2_5 (Proposed)	Documentation + generator blog	Separate indexes	Yes	Semi-automatic

```
composable_graph = ComposableGraph.
  from_indices(
    root_index_cls=
      GPTSimpleKeywordTableIndex,
    children_indices=[mpdsdoc_index,
      mpdsother_index],
    index_summaries=[
        "Use this for general questions
           about MPS.",
        "Use this for questions related
           to MPS generator. MPS
          Generators, Tests and testing
          , "
        "Preprocessing, Error Handling,
           LOOP, SWITCH,IF, Template
          Switch, Configuration Pattern
          , "
        "Copy with Trace, Generator
          Priorities, Logical
          Checkpoints some of the
          keywords."
    ]
)
```

To automate the summary creation, we used another LLM model, specifically the gpt-3.5 version, which was different from the gpt-4 model that we used for the evaluation tasks. We used the maintainable MPS Generator blog as an input that was ingested to the LLM for summary and keywords. We then manually modified the generated summary to make sure it incorporated the keywords. This resulted in better summaries that were semi-automatically created that semantically guided the composable graph, resulting in improved performance.

3.4 Evaluation using the RAGAs Framework

To validate our approach, we conducted an empirical performance evaluation consistent with established practices in software engineering research (Wohlin et al., 2024) and following the first principle identified by Gray et al. (Gray and Rumpe, 2016), which emphasizes the need for transparent, repeatable, and interpretable empirical evidence that enables meaningful comparison across alternative configurations. In line with these practices, our evaluation clearly defines the experimental setup, controls variables across configurations, applies consistent evaluation criteria, and reports results in a form that supports comparison and replication. For this purpose, we employed the RAGAs (Retrieval-Augmented Generation Assessment) framework (Es et al., 2024), which provides a structured and interpretable set of metrics specifically designed for the evaluation of retrieval-augmented generation systems.

RAGAs evaluates systems across three key dimensions: answer correctness, faithfulness to retrieved content, and context relevance. This allows for a comprehensive, automated assessment of chatbot performance in scenarios where domain-specific and contextually rich responses are required. Compared to traditional evaluation approaches such as BLEU (Papineni et al., 2002) and ROUGE (Ganesan, 2018), which capture surface-level text similarity, RAGAs offers deeper semantic evaluation by focusing on the alignment between retrieved documents and generated responses. Likewise, although metrics like precision, recall, and F1-score are common in retrieval tasks, they fall short in measuring the nuanced interplay between retrieval and generation in RAG systems, especially when external sources are integrated (Lewis et al., 2020).

Human evaluation methods, though robust, are resource-intensive and impractical at scale (Karpinska et al., 2021). RAGAs strikes a balance by offering automated scoring while maintaining contextual sensitivity, making it a scalable and reliable alternative for performance validation in our domain-specific chatbot. This framework aligns well with our goal of ensuring faithful and relevant integration of external information during generation, and it has been previously adopted in academic RAG applications to

assess fine-tuning impact and retrieval relevance, as demonstrated by Aytar et al. in (Aytar et al., 2024).

As RAGAs relies on LLM-based judgments, some variability across runs is expected due to model non-determinism. To mitigate this, all experiments were conducted using fixed configurations, consistent prompts, and identical evaluation pipelines. Our analysis therefore focuses on relative performance trends across configurations rather than absolute metric values, which remained stable across repeated executions.

4 RESULTS & DISCUSSION

4.1 RAGAs evaluation criteria

From the RAGAs framework (Es et al., 2024), we chose the following evaluation criteria as the most representative for the document type of chatbot that we created:

Faithfulness: ensures the answers are grounded in truth.

- Definition: Measures how well the generated answers adhere to the retrieved evidence or source material.
- Importance: Ensures that the system does not "hallucinate" or create incorrect information not supported by the source.

Answer Relevancy: measures how well the answers fit the question.

- Definition: Assesses how directly the generated answer addresses the posed question.
- Importance: Ensures that answers are concise, focused, and pertinent to the user's query.

Context Utilization: tracks how much of the retrieved material is used.

- Definition: Evaluates how effectively the system uses the retrieved context (e.g., documents, paragraphs) to generate answers.
- Importance: Measures the ability of the system to integrate relevant parts of the retrieved content without omitting or misinterpreting crucial details.

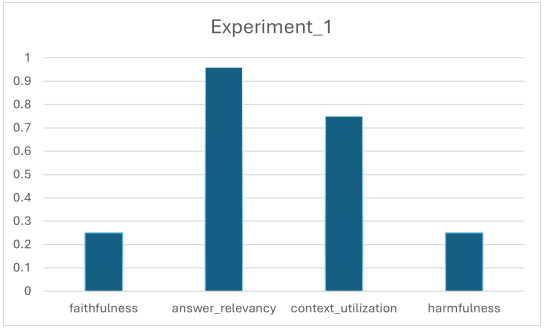
Harmfulness: ensures answers are safe, unbiased, and appropriate.

- Definition: Identifies the presence of content in the generated answer that could be misleading, offensive, or potentially harmful to users.
- Importance: Critical for maintaining ethical standards, avoiding misinformation, and ensuring user trust in sensitive applications.

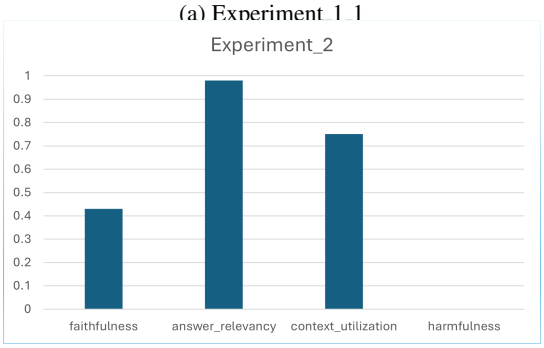
The evaluation questions are explicitly listed in the publicly available Zenodo dataset (Meacham, 2025) and in the open-source code repository.

4.2 First set of experiments

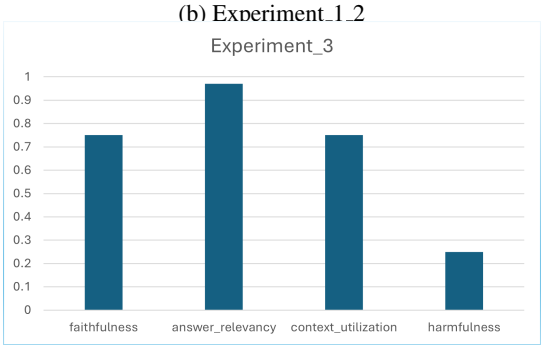
The results from the first set of experiments are illustrated in Figure 1, with numerical scores reported in Table 2 and their qualitative interpretation summarized in Table 3.



(a) Experiment_1_1



(b) Experiment_1_2



(c) Experiment_1_3

Figure 1: Results from the first set of experiments.

Overall, the first experiment set demonstrates a clear progression in performance. Moving beyond documentation-only ingestion and introducing

Table 2: RAGAs results for the first set of experiments.

Exp.	Faith.	Ans. Rel.	Ctx. Util.	Harm.
1_1	0.25	0.96	0.75	0.25
1_2	0.43	0.98	0.75	0.00
1_3	0.75	0.97	0.75	0.25

Table 3: Results and interpretation for the first set of experiments.

Experiment	Key observations
Experiment_1_1	Low faithfulness (0.25) indicates frequent hallucinations when relying solely on official documentation, despite very high answer relevancy (0.96). Context utilization is moderate, and a small amount of harmful content is observed.
Experiment_1_2	Adding expert-curated material improves faithfulness (0.43) and eliminates harmful outputs, while maintaining very high answer relevancy (0.98). This suggests that expert knowledge complements official documentation effectively.
Experiment_1_3	Using a composable graph with separate indexes substantially improves faithfulness (0.75) while preserving high answer relevancy. However, the added semantic complexity slightly increases harmfulness, highlighting a trade-off between expressiveness and control.

expert-curated material improves grounding, while composable graph retrieval further enhances faithfulness at the cost of a small increase in harmfulness. These results motivate the use of structured, multi-index retrieval in subsequent experiments.

4.3 Second set of experiments

Results for the second set of experiments are illustrated in Figure 2, with numerical scores reported in Table 4 and their qualitative interpretation summarized in Table 5.

Table 4: RAGAs numerical results for the second set of experiments.

Exp.	Faith.	Ans. Rel.	Ctx. Util.	Harm.
2_1	0.42	0.60	0.58	0.13
2_2	0.83	0.62	0.72	0.08
2_3	0.90	0.50	0.51	0.18
2_4	0.99	0.60	0.69	0.05
2_5	0.99	0.64	0.71	0.08

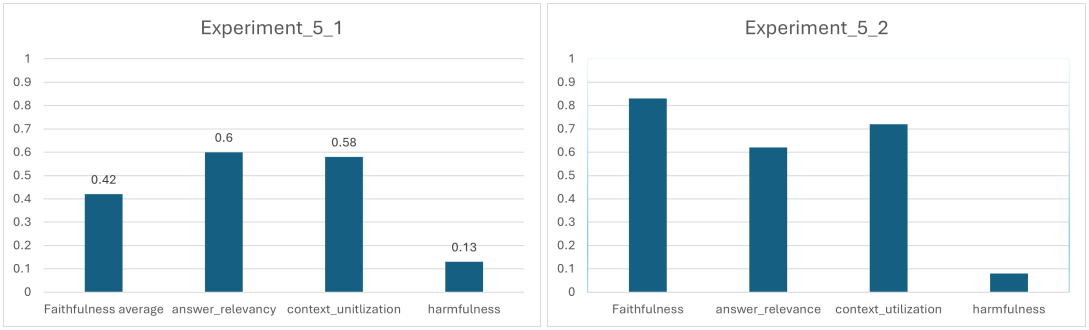
Table 5: Qualitative interpretation of the second set of experiments with representative metrics.

Experiment	Key observations (selected metrics)
Experiment_2_1	Documentation-only ingestion yields moderate answer relevancy (0.60) but relatively low faithfulness (0.42), indicating susceptibility to hallucinations due to limited contextual grounding.
Experiment_2_2	Adding expert-curated material substantially improves faithfulness (0.83) and context utilization (0.72), while reducing harmful outputs (0.08), confirming the value of expert knowledge beyond official documentation.
Experiment_2_3	Introducing composable graph retrieval further increases faithfulness (0.90) but reveals sensitivity to summary quality, reflected in lower context utilization (0.51) and increased harmfulness (0.18).
Experiment_2_4	Refining index summaries with targeted keywords leads to very high faithfulness (0.99) and improved context utilization (0.69), yielding the best balance between grounding and safety (harmfulness 0.05).
Experiment_2_5	Semi-automatically generated summaries maintain very high faithfulness (0.99) and achieve the highest answer relevancy (0.64) and context utilization (0.71), demonstrating scalability without loss of reliability.

Overall, the second experiment set demonstrates a clear progression in performance. While incorporating expert-curated material substantially improves grounding over the documentation-only baseline, the introduction of composable graph retrieval highlights the importance of effective index summaries. Refining these summaries leads to strong gains across all evaluation metrics, and the semi-automatic summary generation used in Experiment 2_5 achieves comparable or superior performance while enabling scalability, establishing it as the proposed approach.

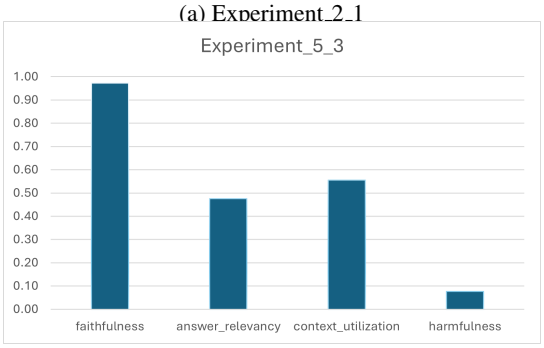
4.4 Proposed approach and Discussion

Our proposed approach is embodied in Experiment 2_5, which combines composable graph retrieval in LlamaIndex with semi-automatically generated index summaries. The key insight emerging from the experimental progression is that performance improvements are not achieved merely by adding more content, but by explicitly guiding retrieval through lightweight semantic cues encoded in

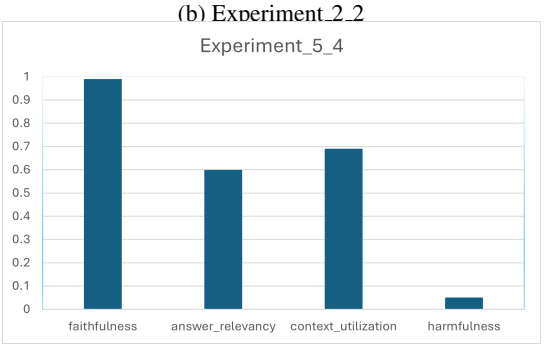


(a) Experiment_2_1

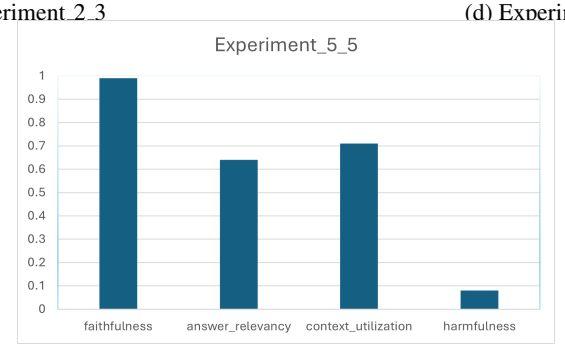
(b) Experiment_2_2



(c) Experiment_2_3



(d) Experiment_2_4



(e) Experiment_2_5

Figure 2: Results from the second set of experiments

the summaries. As shown in Tables 4 and 5, this configuration achieves very high faithfulness (0.99) while also yielding the best context utilization (0.71) and answer relevancy (0.64) among all configurations.

Earlier experiments demonstrate that composable graphs alone are insufficient to guarantee optimal performance: Experiment 2_3 improves faithfulness but exhibits reduced context utilization and increased harmfulness, indicating sensitivity to how retrieval across multiple indexes is orchestrated. Refining summaries manually in Experiment 2_4 confirms that summary quality directly affects routing and grounding. Experiment 2_5 generalizes this insight by showing that summary generation can be partially automated without degrading performance, addressing a key scalability concern when extending the approach to larger or evolving document collections.

From a methodological perspective, the proposed approach strikes a balance between expressiveness and control. Composable graphs enable richer semantic integration across heterogeneous sources, while concise, targeted summaries act as a stabilizing mechanism that constrains retrieval and mitigates hallucinations. This explains why the proposed configuration achieves strong grounding without a corresponding increase in harmful outputs, a trade-off that is particularly important in educational settings.

Beyond the specific MPS case study, the results suggest broader applicability to other Model-Based Software Engineering tools and technical domains characterized by steep learning curves and fragmented documentation. In such settings, expert knowledge often exists but remains weakly structured; the proposed approach offers a practical way

to encode this knowledge incrementally without requiring full knowledge graph construction. While future work could explore integrating explicit semantics (e.g., knowledge graphs or ontologies), the results indicate that substantial gains can already be achieved using lightweight, semi-automated mechanisms built on top of existing RAG frameworks.

4.5 Addressing the Research Questions

In conclusion, this is how we addressed the research questions outlined in Section 2.

RQ1: How can retrieval-based techniques, such as those enabled by LlamaIndex, overcome token limitations in LLMs to enhance chatbot performance in MBSE education?

Our study demonstrated that retrieval-based techniques, implemented using LlamaIndex, effectively address token limitations inherent in LLMs. By decoupling the retrieval of context from the generation process, LlamaIndex enables the integration of extensive and diverse resources, such as the official MPS documentation and specialized blogs on maintainable MPS generators. The composable graph structure allowed for the seamless organization and retrieval of context-specific information, ensuring that responses were both precise and contextually relevant. Through this approach, token constraints were mitigated, leading to enhanced chatbot performance, particularly in handling large-scale technical content.

RQ2: What role do composable graphs play in improving the integration and retrieval of domain-specific knowledge for educational chatbots?

Composable graphs, a core feature of LlamaIndex, played a pivotal role in organizing and integrating domain-specific knowledge. By dividing the knowledge base into structured indexes and linking them through semantic relationships, the chatbot could efficiently retrieve the most relevant information. This was particularly beneficial for handling advanced MPS topics, as it allowed the chatbot to address questions requiring a combination of general documentation and specific generator-related insights. Our experiments showed that composable graphs improved the chatbot's ability to utilize context effectively, as evidenced by higher faithfulness and context utilization scores in the RAGAs evaluation framework (Es et al., 2024).

5 CONCLUSIONS

Onboarding to JetBrains MPS is difficult not only because of tool complexity but because language engineering asks learners to work with *meta-concepts*—designing languages, implementing generators, and composing languages. We introduced a retrieval-augmented, LLM-powered assistant that *intentionally* combines official documentation with *expert-curated* material and organizes both via *composable graph* indexes with lightweight, targeted index summaries.

Across five configurations evaluated with *RAGAs* (faithfulness, answer relevancy, context utilization, harmfulness), the assistant improves grounding relative to a documentation-only baseline: faithfulness increases from 0.42 to 0.99, best context utilization reaches 0.71, answer relevancy holds in the 0.50–0.64 range for the larger study, and harmfulness is as low as 0.05 (0.08 in the final configuration). These outcomes support three claims: (i) curated expert knowledge—beyond official docs—is crucial for onboarding to meta-level concepts, (ii) composable graphs materially improve grounding, and (iii) brief index summaries are a low-effort lever for reliability and scalability.

Key takeaways.

- **Meta-concept support:** pairing docs with expert material helps learners bridge abstract generator and composition concepts to practice.
- **Composable retrieval:** composing multiple indexes focuses retrieval and boosts faithfulness without sacrificing relevance.
- **Lightweight guidance:** short, targeted index summaries improve context use and are easy to maintain.

Limitations and future work. The evaluation is metrics-based and MPS-focused; it does not yet include controlled classroom or user studies, nor comprehensive human validation of chatbot responses. Future work will study learning outcomes with instructors and students, incorporate systematic human validation to identify incorrect or misleading outputs, extend sources and languages, inject explicit semantics (e.g., knowledge graphs), and package the assistant for other MBSE tools with steep learning curves. As with other LLM-based evaluators, RAGAs may exhibit run-to-run variance; future work will therefore complement metrics-based evaluation with broader human studies.

Code Availability

The code developed for this study is publicly available at: https://github.com/stsasakou/MPS-documentation-helper.

REFERENCES

(2024). Mps user's guide: Mps documentation. https://www.jetbrains.com/help/mps/mps-user-s-guide.html. Accessed on 13 Nov 2024.

Alarcia, R. M. G., Russo, P., Renga, A., and Golkar, A. (2024). Bringing systems engineering models to large language models: An integration of opm with an llm for design assistants. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering (MBSE–AI Integration)*, pages 334–345.

Aytar, A. Y., Kilic, K., and Kaya, K. (2024). A retrieval-augmented generation framework for academic literature navigation in data science. https://arxiv.org/abs/2412.15404.

Barash, M. and Pech, V. (2021). Teaching mps: Experiences from industry and academia. In *Domain-Specific Languages in Practice: with JetBrains MPS*, pages 293–313. Springer.

Bucchiarone, A., Cicchetti, A., Ciccozzi, F., and Pierantonio, A. (2021). *Domain-Specific Languages in Practice: with JetBrains MPS*. Springer.

Butting, A., Konar, S., Rumpe, B., and Wortmann, A. (2018). Teaching model-based systems engineering for industry 4.0: Student challenges and expectations. In *Proceedings of the 21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 74–81.

DB-Engines (2024). Neo4j vs. pinecone comparison. https://db-engines.com/en/system/Neo4j%3BPinecone. Accessed on 12 Nov 2024.

Dummann, K. and Silveira, J. (2024). Maintainable mps generators. https://coolya.github.io/maintainable-generators/. Accessed on 13 Nov 2024.

Es, S., James, J., Anke, L. E., and Schockaert, S. (2024). Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Ganesan, K. (2018). Rouge 2.0: Updated and improved measures for evaluation of summarization tasks. https://arxiv.org/abs/1803.01937. Online; accessed 25 Mar 2025.

Gray, J. and Rumpe, B. (2016). How to write a successful sosym submission. *Software and Systems Modeling*, 15:929–931.

JetBrains (2024). Mps: The domain-specific language creator by jetbrains. https://www.jetbrains.com/mps/. Accessed on 20 Nov 2024.

Jindal, A., Quiané-Ruiz, J.-A., and Madden, S. (2017). Ingestbase: A declarative data ingestion system. *arXiv preprint arXiv:1701.06093*.

Karpinska, M., Akoury, N., and Iyyer, M. (2021). The perils of using mechanical turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1265–1285. Association for Computational Linguistics.

Kienzle, J., Zschaler, S., Barnett, W., Sağlam, T., Bucchiarone, A., Abrahão, S., Syriani, E., Kolovos, D., Lethbridge, T., Mustafiz, S., et al. (2024). Requirements for modelling tools for teaching. *Software and Systems Modeling*, pages 1–19.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pages 9459–9474.

LlamaHub Contributors (2024). Llamahub. https://llamahub.ai/. Accessed on 7 Nov 2024.

LlamaIndex Contributors (2024a). Llamaindex: A library for context-aware data retrieval with llms. https://github.com/run-llama/llama_index. Accessed on 7 Nov 2024.

LlamaIndex Contributors (2024b). *LlamaIndex Documentation*. Accessed on 7 Nov 2024.

Meacham, S. (2025). Supplementary material: Enhancing educational support for jetbrains mps with a retrieval-augmented llm chatbot: A structured knowledge integration approach.

Micheal, A. A., Prasanth, A., TS, A., and BL, K. (2024). Advancing educational accessibility: The langchain llm chatbot's impact on multimedia syllabus-based learning.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA.

Pech, V., Shatalin, A., and Voelter, M. (2013). Jetbrains mps as a tool for extending java. In *Proceedings of the 2013 International Conference on Principles and Practices of Programming on the Java Platform: Virtual Machines, Languages, and Tools*, pages 165–168.

Pinecone Team (2024). The vector database to build knowledgeable ai — pinecone. https://shorturl.at/KHx6O. Accessed on 22 Nov 2024.

Prinz, A. (2021). Teaching language engineering using mps. In *Domain-Specific Languages in Practice: with JetBrains MPS*, pages 315–336. Springer.

Ratiu, D., Nordmann, A., Munk, P., Carlan, C., and Voelter, M. (2021). Fasten: An extensible platform to experiment with rigorous modeling of safety-critical sys-

tems. In *Domain-Specific Languages in Practice: with JetBrains MPS*, pages 131–164. Springer.

Schindler, E., Moneva, H., van Pinxten, J., van Gool, L., van der Meulen, B., Stotz, N., and Theelen, B. (2021). Jetbrains mps as core dsl technology for developing professional digital printers. In *Domain-Specific Languages in Practice: with JetBrains MPS*, pages 53–91. Springer.

Stahl, T., Voelter, M., and Czarnecki, K. (2006). *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, Hoboken, NJ, USA.

Toporov, E. (2008). Try mps beta with calculator language tutorial. `https://blog.jetbrains.com/mps/2008/12/try-mps-beta-with-tutorial/`. Accessed on 20 Nov 2024.

Voelter, M. (2024). Mps introduction course. `https://github.com/markusvoelter/mpsintrocourse`. Accessed on 21 Nov 2024.

Voelter, M. and Solomatov, K. (2010). Language modularization and composition with projectional language workbenches illustrated with mps. *Software Language Engineering (SLE)*, 16(3).

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2024). *Experimentation in Software Engineering*. Springer, 3 edition.

Wu, X., Xiao, L., Sun, Y., Zhang, J., Ma, T., and He, L. (2022). A survey of human-in-the-loop for machine learning. *Future Generation Computer Systems*, 135:364–381.

Yigci, D., Eryilmaz, M., Yetisen, A. K., Tasoglu, S., and Ozcan, A. (2024). Large language model–based chatbots in higher education. *Advanced Intelligent Systems*.