Teaching UML using a RAG-based LLM

Pasquale Ardimento
Dept. of Computer Science
University of Bari Aldo Moro
Bari, Italy
pasquale.ardimento@uniba.it

Mario Luca Bernardi
Dept. of Engineering
University of Sannio
Benevento, Italy
bernardi@unisannio.it

Marta Cimitile
Dept. of Law and Economics
UnitelmaSapienza University
Rome, Italy
marta.cimitile@unitelmasapienza.it

2024 International Joint Conference on Neural Networks (IJCNN) | 979-8-3503-5931-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/IJCNN60899.2024.10651492

Abstract—**Teaching the Unified Modelling Language (UML) is a critical task in the frame of Software Engineering courses. Teachers need to understand the students' behavior along with their modeling activities to provide suggestions and feedback to avoid more frequent mistakes and improve their capabilities. This paper presents a novel approach for teaching the UML in Software Engineering courses, focusing on understanding and improving student behavior and capabilities during modeling activities. It introduces a cloud-based tool that captures and analyzes UML diagrams created by students during their interactions with a UML modeling tool. The key aspect of the proposal is the integration of a Retrieval Augmented Generation Large Language Model (RAG-based LLM), which generates insightful feedback for students by leveraging knowledge acquired during the modeling process.**

The effectiveness of this method is demonstrated through an experiment involving a substantial dataset comprising 5,120 labeled UML models. The validation process confirms the performance of the UML RAG-based LLM in providing relevant feedback related to entities and relationships in the students' models. Additionally, a qualitative analysis highlights the user satisfaction, underscoring its potential as a valuable tool in enhancing the learning experience in software modeling education.

Index Terms—**Deep Learning, Generative AI, LLMs, Computing Education, UML, Software Modelling**

I. Introduction

Teaching UML [6] design is an important activity in Software Engineering (SE) courses since it represents the de-facto standard for software modeling [22] and it is very useful to understand the SE basic concepts [25]. However, UML is complex and it is hard to grasp for novice students who are always prone to make the same mistakes [7]. Great support to the students can be given by several UML teaching tools [22] that can help and drive the novices to build and manipulate the UML diagrams [20]. These tools and approaches have also recently been introduced to support teachers in the student's monitoring and evaluation activities [23].

However, on the base of our knowledge, the existing teaching UML approaches, fail to exploit the the current advantages of the Large Language Models (LLMs) [16] to assist the novice modelers in their activities. LLMs are deep neural networks trained on large amounts of data to learn undiscovered patterns and relationships in natural language text [16]. According to their characteristics, their impact on education fields is substantial, offering a wide set of applications and opportunities to enhance learning and teaching experiences [13]. From the teacher's point of view, LLMs provide valuable support in preparing and executing educational tasks, as well as evaluating students' assignments [17] [26]. For instance, they prove effective in implementing scaffolding approaches by offering timely feedback and support during learning tasks [4]. Even if the LLMs applications in software model development show not good results [8], we suppose that LLMs can play an important role in teaching software modeling and more in general to the software modeling community. This paper introduces a novel approach that uses a Retrieval-Augmented Generation (RAG) framework to capture students' modeling products when he/she interacts with the Visual Paradigm environment and gives useful feedback to the students and the teachers of the performed modeling activities. The proposed tool uses RAG [19] combined with LLaMA Large Language Models (LLM) [27] and leverage domain-specific embedding models to feed the retrieval stage. The LLM allows for generating useful feedback about how the modelers can improve or correct their behavior. The LLM-generated feedback reports also describe students' mistakes and incorrect behavior which can be useful to improve their modeling capabilities and reduce the teacher's effort. The RAG-based LLM is an extension of the Eclipse Che cloud IDE. This allows for performing student monitoring in a non-intrusive way without interrupting their modeling activities. In this paper, we show an empirical validation of the proposed approach in a real context. The document is structured as follows: in Section II the most relevant related works are reported, Section III concerns the pre-trained neural networks adopted, some fundamental concepts are in fact explained, and Section IV describes the approach used in detail. The description of the experiment is in Section V, and Section VI shows the results obtained. Finally, Section VII and VIII report threats to validity and conclusions, respectively.

II. Related Work

Computing education has been strongly influenced in the last years by innovative solutions designed to support students in several coding and modeling activities [10]. In particular, the more recent advances in artificial intelligence show a great capability of AI-based tools to evaluate students' code [3] by producing a report of student's mistakes and misconceptions. The mining of students' behavior in code development activities is also proposed in [2], [5] where a Process Mining based approach is adopted. Referring to the coding activities, several tools support students in the generation of source code starting from the problem description written with natural language [10]. Despite the inspiring results of AI-based tools in teaching coding activities, in this study, we will focus on UML modeling teaching. The UML teaching is actually supported by a limited number of tools [1], [9], [14]. Some of these approaches are limited to only support the creation of class diagrams [9], [12] and are further to include AI-based functionalities. The creation of a more complete and consistent UML model is also supported by SD4ED [1]. This approach provides instant feedback while students are creating the model based on the specifications of the underlying model and the past messages. The tool presented in [14] introduces the AutoER system that allows the automatic evaluation and generation of UML database models. It provides the student feedback through the question answering. Differently from the existing tools, the UML RAG-based LLM focuses on the use of LLM solutions to answer the student questions and give them feedback according to their modeling products. The adoption of a repository that collects all the UML artifacts produced by the students also allows to compare different modeling executions on the UML by identifying behavioral similarities and differences.

III. Background

A. Large Language Models and RAG

LLMs are sophisticated deep neural networks trained on extensive datasets to discern patterns and relationships within natural language corpora [24]. This enables LLMs to closely emulate human proficiency in executing language-related tasks, including translation, summarization, and question answering, achieving remarkable accuracy [18]. Nonetheless, while LLMs exhibit grammatical correctness in the generated texts, they frequently fall short of delivering extreme accuracy or suitability for specific contexts [24], [16]. LLMs heavily rely on extensive input data for training, and the quality of their output is significantly influenced by the characteristics of the input data. Consequently, biases or errors present in the training set can undergo significant amplification when passing through LLMs [15]. A promising solution to address these issues, especially in knowledge-intensive tasks, is RAG [19]. This approach involves integrating specialized and dynamic external repositories with the original knowledge base used for LLM training [19]. In-Context Learning (ICL) [11] facilitates the external retrieval of relevant information using suitable search algorithms, which is then incorporated into the LLM, providing additional context-related data [15].

The external repository is typically enhanced based on the specific needs of certain domains and can be regularly updated with new knowledge, improving the precision of the model's output [15]. RAG typically consists of a retriever and a generator and involves three main steps: retrieve, augment, and generate [19].

In the retrieval step, a user query is utilized to fetch relevant context text documents (from an external knowledge base) with parameters. The query is embedded into a vector space, and the closest documents are extracted based on vector similarities. In the augmentation step, a prompt template is formed by combining the initial query and the additional context. In the generating step, the LLM processes the retrieval-augmented prompt to generate an answer, leveraging contextual information from the retrieved chunks.

B. LLaMA

This study uses an LLaMA as an LLM. LLaMA [27] is a collection of foundational extensive language models ranging from 7 billion to 70 billion parameters.

These models undergo training on an exceptionally vast number of samples sourced from public datasets, achieving superior performance compared to alternative methodologies. As an illustration, for the 70 billion parameters model, recent investigations [27] indicate that LLaMA competes with Chinchilla or PaLM-540B which are the leading large language models. The training process employs large transformers, utilizing a standard optimizer, and enhances training stability by normalizing the input of each transformer sub-layer. The training datasets exhibit considerable heterogeneity, encompassing various domains and incorporating all publicly accessible datasets utilized in extensive language model training. In this study the different variants of LLaMA will be adopted and evaluated: LLaMA 7b, LLaMA 13b, LLaMA 70b.

IV. The UML RAG-based LLM

The proposed UML RAG-based LLM establishes a supervised modeling environment, systematically documenting all modeling activities and collecting information about the produced UML diagrams. The diagram information is then sent to a repository and used to obtain text embedding models sent to the RAG Vectore Store. The Vector Store receives the query from the retrieves and generates context data extracting knowledge from the Vector store. The retriever receives the prompt from the student (or the teacher) and creates the query that is sent to the Vectore Store. Moreover, the retriever also receives the context information generated by the Vectore store and uses the prompt, the query, and the context information as input for the LLM. The LLM sent the feedback to the student (or the teacher). Fig. 1 illustrates the comprehensive architecture of the system, encompassing key components: Diagram Information, Rag Component, and LLM. As depicted in Fig. 1, the tool prioritizes versatility in its application. In the following subsections, a further description of each component is reported.

A. UML Info

The UML Info, a plugin for Visual Paradigm environment, is capable of generating a textual representation of UML diagrams based on their graphical counterparts. It can extract and organize information about UML elements, attributes, operations, and relationships defined within the diagram. The resulting text provides a clear and structured overview of the diagram's elements, their properties, and their interconnections.

In more detail, the Diagram Info represents the modeling properties made across all UML diagrams. It operates in a manner similar to a daemon because: i) it runs unattended, ii) is constantly in the background, iii) is available at all times, iv) starts when Visual Paradigm starts and runs until Visual Paradigm stops. Importantly, it never interrupts or hinders modeling activities.

For each "Visual Paradigm" project, UML Info generates a comprehensive information report encompassing all the diagrams within the project. The diagram information represented in a report can be categorized as reported below:

- **Project Property:** Lists all modeling properties issued to a Visual Paradigm project (e.g., project name, UML version, author, company, project description).
- **Diagram Property:** Lists all modeling properties issued to a UML diagram (e.g., diagram name, parent model, diagram description).
- **Model Element Property:** Lists all modeling properties defined for a property of a UML model element contained in a diagram. Each model element has its properties, and these properties can be atomic or composite. To provide an example, the visibility of a class represents an atomic property, whereas a class attribute serves as a composite property due to its characteristics such as name, visibility, type modifier, multiplicity, and more. The name property and its corresponding value are consistently recorded, and depending on the specific property that has been modified, additional information may be captured. Atomic properties are inherently associated with a model element and assume a default value if not explicitly specified. On the other hand, composite properties are explicitly added to a model element, and as a result, the tool also records their respective IDs. Examples of UML model elements are classes, use cases, actors, packages, templates, frames, and more.
- **Models Relationship Property:** Lists all modeling properties issued to a relationship between UML model elements. The plugin records specific properties based on the relationship type and the elements involved, such as association, aggregation, generalization between classes, includes and extends between use cases, and more. For each relationship, the plugin stores its ID and the ID of the originating model element, and the ID of the element where the relationship terminates.
- **Extension Mechanism Property:** Lists all modeling properties associated with an extension mechanism (e.g., stereotypes and tagged values) for a UML model element. Since any model event could have multiple stereotypes or tagged values, the tool captures and categorizes them accordingly.

It is important to note that the current version is only capable of generating textual descriptions for class diagrams. The Diagram Info plugin is downloadable starting from https://sites.google.com/view/uml-info/.

B. The RAG-based LLM

The RAG-based LLM relies on a specialized model designed for the UML domain, enabling the generation of text embeddings in accordance with the UML diagram information. These produced embeddings are then transmitted to the RAG vector repository. The RAG repository serves as a compilation of vectors characterizing the explored domain of interest based on the Diagram information extracted by the UML Info plugin. Subsequently, the embeddings are forwarded to the retriever component, which also takes in the user prompt. The retriever calculates the cosine similarity between the student (or teacher) prompt and the vectors, generating the top k-reports (i.e., summary, explanations and suggestions) based on the highest similarity scores. The top k-reports are employed by the LLM as contextual information to enhance the generative model, ensuring a more precisely generated report in response to the student prompt it received as input. In summary, the standard procedure for executing a question-answering task is the following:

- Creation of the Vector Store. This involves producing embedding vectors for each document in the local knowledge base. These documents are then arranged in a Vector Store, using the embedding vectors as an indexing mechanism.
- Context retrieval. The query is embedded using the same technique applied to the information. This embedded query is then used to pinpoint and retrieve the most relevant information, according to

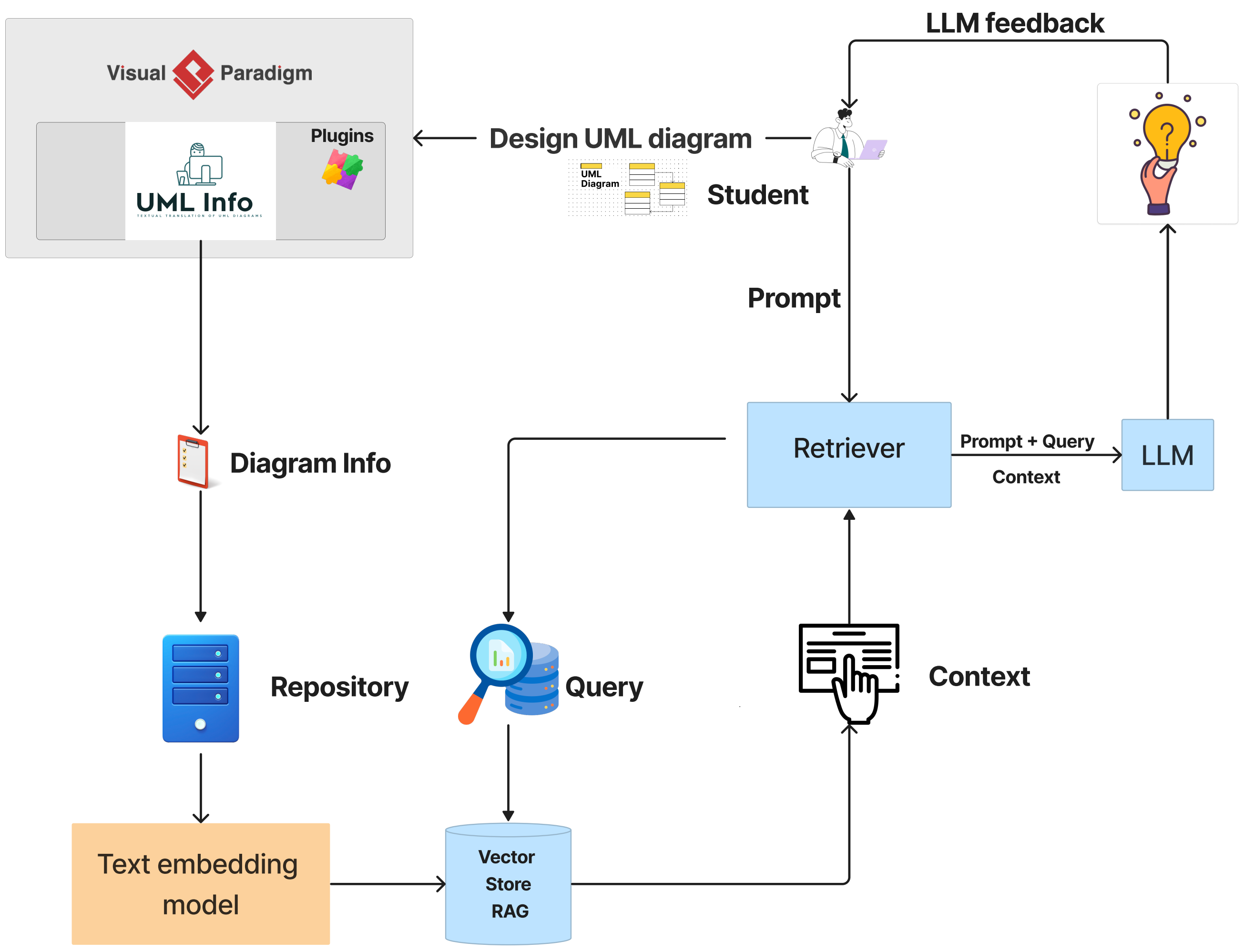

Fig. 1: The UML RAG-based LLM architecture.

the adopted embedding model, from the Vector Store, offering the necessary context.

- Feeding the LLM. The chosen relevant context information, along with the input query and the prompt are input into the LLM. This fusion empowers the LLM to generate a response specifically tailored to the information contained within the local knowledge base.

V. Experiment Description

This section describes the empirical validation of the UML RAG-based LLM approach. The aim is to evaluate the effectiveness of the UML RAG-based LLM in identifying the correct entities and relationships within the provided diagrams.

A. *Experimental setting*

To evaluate the effectiveness of a RAG system in enhancing UML diagrams through feedback and suggestions, we designed a case study covering different perspectives. This RAG-based system merges retrieval mechanisms with generative models, utilizing a database of UML diagrams knowledge, previous modeled examples, and best practices. When presented with a new UML diagram, the system retrieves relevant knowledge and similar examples and generates its feedback for the user. This kind of system requires a twofold validation based on qualitative and quantitative aspects.

More precisely, the quantitative validation covers the correctness (in terms of precision and recall) of the answers with respect to the entities and relationships that are both in the model provided by the user and in the LLM answer.

Conversely, the qualitative validation is based on answer evaluation from a user-effectiveness standpoint. This part has been performed by questionnaire analysis given to practitioners to quantify their satisfaction with the provided support. We elaborate on both these aspects formalizing the process.

Element Type	Mean	Std. Dev.
Elements	128.13	126.84
Components	0.21	0.73
Packages	1.08	1.06
Classes	4.71	6.47
Enums	0.23	0.71
Datatypes	0.63	2.67
Properties	22.63	24.49
Operations	7.15	15.87
Generalizations	1.84	4.09
Associations	6.93	7.78

TABLE I: Element Types: Average and Std. Dev.

For quantitative validation, participants submit UML diagrams, which UML experts then annotate to identify correct entities and relationships. The effectiveness of the system is evaluated based on correctness (accuracy in identifying correct and incorrect elements), precision (ratio of relevant suggestions to total suggestions), and recall (the system's ability to recognize all pertinent feedback points). These metrics are aggregated across all diagrams, and statistical tests, such as t-tests, are used to assess their significance. Fig. 2 shows the categories of the 422 selected diagrams.

In parallel, qualitative validation is conducted through a questionnaire designed to capture the effectiveness and user satisfaction with the system's feedback. This questionnaire comprises both closed and open-ended questions. After interacting with the system, 12 participants complete this questionnaire, providing insights into their user experience, perceived usefulness of the feedback, and suggestions for improvement. Responses are analyzed using thematic analysis to identify common themes and patterns, and satisfaction levels are quantified using Likert-scale responses. The questionnaire responses are shown in Table III.

The results are then interpreted to provide a comprehensive overview of the system's performance. Quantitative results focus on aggregate correctness, precision, and recall scores, offering a clear picture of the system's efficiency in UML diagram improvement. Qualitative results summarize key themes from user feedback, highlighting strengths and areas needing improvement.

The conclusion drawn from the study focuses on the overall utility of the RAG-based system in aiding UML diagram enhancement, discussing its potential applications in both educational and professional settings. Based on the outcomes of this experiment, recommendations for system improvements are suggested, along with proposals for further studies to continue validating and enhancing the system's capabilities.

B. Dataset

For the quantitative evaluation, ModelSet [21] dataset is used. This dataset is a collection of software models created to facilitate research in Machine Learning (ML) and Model-Driven Engineering (MDE). It includes 5,466 Ecore and 5,120 UML-labeled models. The dataset is organized in a way that allows for easy manipulation using SQLite databases, with tables for models, metadata, and statistics. The metadata table contains labels entered by the user who performed the labeling, and these labels are also available in JSON format. This structure enables direct interaction with the SQLite database for exploratory queries. The dataset also includes statistics about the models, such as the number of elements and various types of meta-elements like classes, attributes, and packages. ModelSet is accompanied by a Python library that facilitates its use with standard machine learning libraries, making it a valuable resource for building classifiers and other ML applications in the context of software modeling.

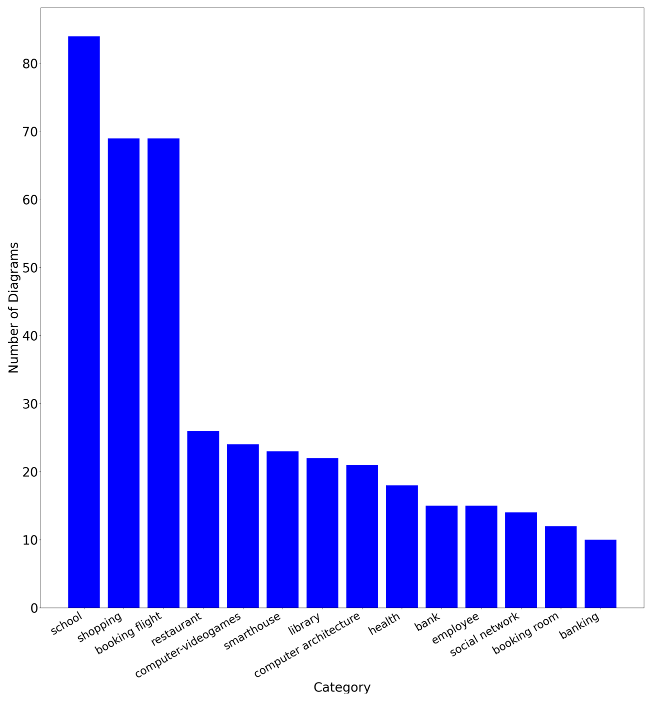

Fig. 2: Number of diagrams selected for the experiment.

VI. Results and Discussion

For the quantitative evaluation, the performance of the evaluated LLaMA models on the ModelSet is reported in Table II. The table provides a detailed comparison of various elements across different models and configurations, specifically LLaMA 7b, LLaMA 13b, and LLaMA 70b, each tested in 'Plain' and 'RAG' settings. The metrics used for comparison are Precision, Recall, and F1 score with respect to correct/uncorrect items found in both the diagram and the answer. The rows of the table report the UML element considered in this study coherently to Table I. Table II highlights that for

TABLE II: Performance Metrics of LLaMA Models in Plain and RAG Variants

Element	**LLaMA 7b**						**LLaMA 13b**						**LLaMA 70b**					
	Plain			**RAG**			**Plain**			**RAG**			**Plain**			**RAG**		
	Prec.	**Recall**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**	**Prec.**	**Recall**	**F1**
Elements	0.12	0.20	0.15	0.49	0.52	0.50	0.17	0.24	0.20	0.65	0.63	0.64	0.25	0.25	0.25	0.70	0.72	0.71
Components	0.02	0.10	0.03	0.59	0.61	0.60	0.05	0.19	0.08	0.68	0.70	0.69	0.10	0.26	0.15	0.73	0.77	0.75
Packages	0.15	0.17	0.16	0.61	0.58	0.59	0.21	0.19	0.20	0.69	0.69	0.69	0.24	0.28	0.26	0.71	0.76	0.73
Classes	0.21	0.19	0.20	0.66	0.51	0.58	0.22	0.22	0.22	0.71	0.60	0.65	0.29	0.26	0.28	0.77	0.62	0.69
Enums	0.09	0.11	0.10	0.69	0.59	0.64	0.13	0.16	0.14	0.72	0.68	0.70	0.18	0.21	0.20	0.74	0.75	0.74
Datatypes	0.13	0.14	0.13	0.62	0.68	0.65	0.16	0.24	0.19	0.70	0.79	0.74	0.24	0.32	0.28	0.77	0.85	0.81
Properties	0.12	0.01	0.02	0.56	0.51	0.53	0.19	0.11	0.14	0.67	0.62	0.64	0.27	0.13	0.17	0.70	0.69	0.70
Operations	0.11	0.22	0.15	0.68	0.72	0.70	0.24	0.28	0.26	0.71	0.81	0.76	0.34	0.36	0.35	0.79	0.83	0.81
Generalizations	0.06	0.09	0.07	0.38	0.43	0.40	0.14	0.19	0.16	0.54	0.52	0.53	0.16	0.23	0.19	0.57	0.60	0.58
Associations	0.01	0.02	0.01	0.31	0.43	0.36	0.12	0.22	0.16	0.49	0.54	0.51	0.14	0.26	0.18	0.54	0.57	0.55
Averages	**0.10**	**0.13**	**0.10**	**0.56**	**0.56**	**0.56**	**0.16**	**0.20**	**0.17**	**0.66**	**0.66**	**0.66**	**0.23**	**0.26**	**0.24**	**0.70**	**0.72**	**0.71**

TABLE III: Participant Questionnaire Responses (Measured on a Likert Scale)

No.	**Expertise Level**	**Relevance of Concepts**	**Helpfulness**	**Accurate answers**	**Willingness to Use**
1	Expert	3	3	3	3
2	Expert	3	3	3	3
3	Student	4	4	4	4
4	Expert	3	3	3	3
5	Naive	5	5	3	5
6	Expert	3	3	3	3
7	Student	3	4	3	4
8	Naive	5	5	3	3
9	Student	4	3	4	4
10	Naive	5	4	3	4
11	Naive	5	5	4	4
12	Student	4	5	4	4

TABLE IV: Statistical Test Results Including Relevance

Metric	**t-statistic**	**mean**	**p-value**
Usability Rating	3.52	3.92	0.0047
Helpfulness of Feedback	3.53	3.91	0.0049
Accuracy of Suggestions	2.33	3.33	0.0038
Relevance of Concepts	3.55	3.67	0.0046

the LLaMA 7b model, the ’RAG’ configuration generally shows significantly better performance than the ’Plain’ setting across all elements. This is evident in the higher Precision, Recall, and F1 scores in the ’RAG’ setting. For example, for ’Elements’, the F1 score improves from 0.15 in ’Plain’ to 0.50 in ’RAG’. This is consistent across all metrics also for LLaMa 13. Notably, ’Components’ and ’Operations’ show a substantial increase in performance in the ’RAG’ configuration. Moreover, concepts are often missed by the LLaMa plain whereas are much better recognized by the RAG version that is augmented with UML knowledge. Hence as a general trend, it is confirmed that ’RAG’ consistently offers better Precision, Recall, and F1 scores compared to the ’Plain’ setting. However the overall performance also seems to improve with the size of the model, as seen in the progression from LLaMA 7b to LLaMA 70b, but they never become competitive with RAG which has a greater impact with respect to size. Moreover, the difference is more pronounced in the larger models (i.e., LLaMA 13b and LLaMA 70b), indicating a scaling effect where larger models benefit more from the ’RAG’ architecture.

Concerning the qualitative analysis, to test whether the system is perceived as better than the average, we can use a one-sample t-test. This test compares the mean of the sample data to a known value (in this case, the ’average’ value). In all cases, as reported in Table IV, the low p-values (< 0.05) suggest that there is significant evidence to reject the null hypothesis that the ratings are neutral. These results could indicate that the system is performing better than an average level in terms of usability, helpfulness, and accuracy, as perceived by the users. Specifically, we state the following findings from the user perspective satisfaction standpoint:

- **Usability Rating**: The t-statistic is 3.52, with a mean of 3.92 and a p-value of 0.0047. This indicates that there is a statistically significant difference from the neutral response, suggesting that the participants’ perception of usability is significantly positive.
- **Helpfulness of Feedback**: A t-statistic is 3.53, with a mean of 3.91 and a p-value of 0.0049 means that the helpfulness of the feedback also does significantly differ from the neutral mean. This implies that the feedback’s helpfulness is perceived as better than average.
- **Accuracy of Suggestions**: The t-statistic for the accuracy of suggestions is 2.33 with a mean of 3.33 and a p-value of 0.0038. This indicates a significant small deviation from neutral, suggesting that the system’s suggestions are perceived as a bit more

accurate with respect to neutral.

- **Relevance of Concepts**: The t-statistic for the relevance of concepts in the system's answers is 3.55 with a mean of 3.67 and a p-value of 0.0046. Similar to the other metrics, this indicates that there is a statistically significant difference from a neutral response. This suggests that the relevance of the system's concepts is perceived as better than average by the participants.

VII. Threats to the validity

Some concerns about the validity of the study should be noted.

A primary concern for validity lies in the use of open LLM models. The use of closed models like ChatGPT in a research setting is impractical due to flexibility issues, their high costs, and the inability to control their development.

The swift advancement of LLM models poses an internal validity concern. The ongoing rapid developments in LLM technology may quickly render our results outdated. To counter this, in this work, we compare recent open LLM models of the LLaMA family.

Regarding external validity, the difficulty in replicating the experiments is a significant issue. This is because LLMs can produce varied responses to identical prompts and there is very limited control over this. To overcome this issue we use several strategies like the use of standardized prompts where possible. This reduces variability in responses due to differences in how questions or commands are phrased. Moreover, we performed multiple runs conducting the same experiment multiple times and average the results. This can help smooth out anomalies due to the stochastic nature of LLM responses.

Another point concerning external validity is the limited generalizability of our results. Our study concentrates on a dataset specific to the aviation industry, and we recognize that broader generalization and validation of our findings will be a focus in future research.

VIII. Conclusions

In this paper, we have introduced and evaluated a RAG based system designed to assist students and modelers in enhancing UML diagrams. Our system leverages advanced RAG methodologies to provide insightful feedback and suggestions, aiming to improve the quality and accuracy of UML diagrams.

The validation of our system was conducted through a twofold approach: quantitative and qualitative assessments. Quantitatively, we focused on measuring the correctness, precision, and recall of the systems responses concerning the identification of accurate entities and relationships in UML diagrams. These metrics were crucial in determining the system's capability to accurately interpret and provide relevant feedback on complex diagrams. The results from this quantitative analysis demonstrated the systems robustness in understanding and analyzing UML diagrams, affirming its effectiveness in a technical context. On the qualitative side, we employed a comprehensive questionnaire to gather feedback from practitioners. This approach was pivotal in assessing the systems effectiveness from the user's perspective, providing insights into its usability, applicability, and overall impact on the modeling process. The feedback from this analysis highlighted the system's utility in real-world scenarios, showcasing its potential to significantly aid in the learning and application of UML modeling.

For future work we plan to expanding the systems capabilities to cover a broader range of diagram types and modeling languages could greatly enhance its applicability. Secondly, integrating more sophisticated natural language processing techniques could improve the system's ability to understand and respond to user queries more contextually. Lastly, continuous updates and refinements based on user feedback will be vital in ensuring the system remains relevant and effective against the backdrop of rapidly evolving modeling tools and methodologies.

Appendix A
Questionnaire for Qualitative Assessment

This appendix presents the questionnaire used for the qualitative assessment of our study. Participants were asked to complete this questionnaire after interacting with the system.

Instructions:

- Please answer the following questions based on your experience with the system.
- All responses will remain confidential and will be used solely for the purpose of this research.

A. *Participant Information*

1. Occupation:
2. Level of expertise with UML Diagrams (Naive/Student/Expert):

B. *Quality of Feedback*

3. On a scale of 1-5, how would you rate the overall usability of the system? (1: Very Difficult, 5: Very Easy)
4. How concepts in the system's answer are relevant to entities and relationships contained in the diagram ? (1: Not Relevant, 5: Very Relevant)
5. How helpful was the feedback provided by the system? (1: Not helpful, 5: Very Helpful)
6. Did the system provide new insights or perspectives on your UML diagram? Please provide examples.
7. How would you rate the accuracy of the suggestions made by the system? (1: Very Inaccurate, 5: Very Accurate)

C. *Overall Experience*

8. Would you use this system again for UML diagram analysis? (Not at all /Weakly no / Weakly yes / Absolutely yes):
9. Additional comments or feedback:

References

[1] Sohail Alhazmi, Charles Thevathayan, and Margaret Hamilton. Learning UML sequence diagrams with a new constructivist pedagogical tool: SD4ED. In Mark Sherriff, Laurence D. Merkle, Pamela A. Cutter, Alvaro E. Monge, and Judithe Sheard, editors, *SIGCSE '21: The 52nd ACM Technical Symposium on Computer Science Education, Virtual Event, USA, March 13-20, 2021*, pages 893–899. ACM, 2021.

[2] Pasquale Ardimento, Lerina Aversano, Mario Luca Bernardi, Vito Alessandro Carella, Marta Cimitile, and Michele Scalera. UML miner: A tool for mining UML diagrams. In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, MODELS 2023 Companion, Västerås, Sweden, October 1-6, 2023*, pages 30–34. IEEE, 2023.

[3] Pasquale Ardimento, Mario Luca Bernardi, and Marta Cimitile. Software analytics to support students in object-oriented programming tasks: An empirical study. *IEEE Access*, 8:132171–132187, 2020.

[4] Pasquale Ardimento, Mario Luca Bernardi, Marta Cimitile, and Giuseppe De Ruvo. Mining developer's behavior from web-based ide logs. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 277–282, 2019.

[5] Pasquale Ardimento, Mario Luca Bernardi, Marta Cimitile, and Giuseppe De Ruvo. Mining developer's behavior from web-based IDE logs. In Sumitra Reddy, editor, *28th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2019, Naples, Italy, June 12-14, 2019*, pages 277–282. IEEE, 2019.

[6] Grady Booch, James Rumbaugh, and Ivar Jacobson. *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional, 2005.

[7] Philip J. Burton and Russel E. Bruhn. Using uml to facilitate the teaching of object-oriented systems analysis and design. *J. Comput. Sci. Coll.*, 19(3):278290, jan 2004.

[8] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml. *Software and Systems Modeling*, 22(3):781–793, 2023.

[9] Eric Crahen, Carl Alphonce, and Phil Ventura. Quickuml: A beginner's uml tool. In *Companion of the 17th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '02, page 6263, New York, NY, USA, 2002. Association for Computing Machinery.

[10] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. Computing education in the era of generative ai, 2023.

[11] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. A survey on in-context learning, 2023.

[12] Dimitris Dranidis, Ioanna Stamatopoulou, and Marina Ntika. Learning and practicing systems analysis and design with studentuml. In *Proceedings of the 7th Balkan Conference on Informatics Conference*, BCI '15, New York, NY, USA, 2015. Association for Computing Machinery.

[13] Benedikt Fecher, Marcel Hebing, Melissa Laufer, Jörg Pohle, and Fabian Sofsky. Friend or foe? exploring the implications of large language models on the science system. *AI & SOCIETY*, 2023.

[14] Sarah Foss, Tatiana Urazova, and Ramon Lawrence. Learning UML database design and modeling with autoer. In Thomas Kühn and Vasco Sousa, editors, *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS 2022, Montreal, Quebec, Canada, October 23-28, 2022*, pages 42–45. ACM, 2022.

[15] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2024.

[16] Philipp Hacker, Andreas Engel, and Marco Mauer. Regulating chatgpt and other large generative ai models, 2023.

[17] Jaeho Jeon and Seongyong Lee. Large language models in education: A focus on the complementary relationship between human teachers and chatgpt. *Education and Information Technologies*, 2023.

[18] Enkelejda Kasneci, Kathrin Sessler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, Stephan Krusche, Gitta Kutyniok, Tilman Michaeli, Claudia Nerdel, Jürgen Pfeffer, Oleksandra Poquet, Michael Sailer, Albrecht Schmidt, Tina Seidel, Matthias Stadler, Jochen Weller, Jochen Kuhn, and Gjergji Kasneci. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274, 2023.

[19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[20] Antonio Vallecillo Loli Burgueño and Martin Gogolla. Teaching uml and ocl models and their validation to software engineering students: an experience report. *Computer Science Education*, 28(1):23–41, 2018.

[21] José Antonio Hernández López, Javier Luis Cánovas Izquierdo, et al. Modelset: a dataset for machine learning in model-driven engineering. *Softw. Syst. Model.*, 21(3):967986, jun 2022.

[22] Yuting Lu and Cristina Adriana Alexandru. Systematic review of uml diagramming software tools for higher education software engineering courses. In *Proceedings of the 2023 Conference on United Kingdom & Ireland Computing Education Research*, UKICER '23, New York, NY, USA, 2023. Association for Computing Machinery.

[23] Salisu Modi, Hanan Taher, and Hoger Mahmud. A tool to automate student uml diagram evaluation. *Academic Journal of Nawroz University*, 10:189–198, 09 2021.

[24] Ipek Ozkaya. Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE Software*, 40(3):4–8, 2023.

[25] Rebecca Reuter, Theresa Stark, Yvonne Sedelmaier, Dieter Landes, Jürgen Mottok, and Christian Wolff. Insights in students problems during uml modeling. In *2020 IEEE Global Engineering Education Conference (EDUCON)*, pages 592–600, 2020.

[26] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*. ACM, aug 2022.

[27] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.