A comparison of different Large Language Models for the generation of UML class diagrams

A comparison of different Large Language Models for the generation of UML class diagrams

1st Giacomo Garaccione
Politecnico di Torino
Torino, Italy
giacomo.garaccione@polito.it

2nd Diego Maria Calabrese
Politecnico di Torino
Torino, Italy
diegomaria.calabrese@studenti.polito.it

3rd Riccardo Coppola
Politecnico di Torino
Torino, Italy
riccardo.coppola@polito.it

4thLuca Ardito
Politecnico di Torino
Torino, Italy
luca.ardito@polito.it

***Abstract*—Large Language Models (LLMs) have emerged in recent years as an effective technology for various software related applications such as requirements definition, code generation and analysis, and software testing, thanks to their effective text analysis capabilities. Among these activities, the automated generation of Unified Modeling Language (UML) class diagrams is a field that has been explored with varying levels of success. This paper builds on previous work comparing human-made and LLM-generated diagrams by evaluating the performance of four state-of-the-art LLMs, including both proprietary and open-source models, in the generation of class diagrams as solutions of university exercises defined via natural language specifications. By using role-based few-shot prompting strategies, we generated class diagrams and evaluated them according to standard evaluation frameworks focusing on adherence to syntactic rules, semantic accuracy, and completeness with respect to the exercises' domain. The results show that all models displayed a reasonable capability to generate sufficiently complete diagrams, although with differences in their strengths and weaknesses: proprietary models (namely, ChatGPT and Gemini) excelled in completeness (avg. 64.2% for ChatGPT) and syntax quality (0 errors for both on all the exercises), DeepSeek proved to be the best in following semantic constraints (avg. 3 errors), and Qwen, while achieving similar completeness scores to the other models, struggled with following syntactic rules. These findings highlight the potential of LLMs as educational modeling assistants with their varying degrees of competence, suggesting future benefits with their integration in educational tool-based modeling environments.**

***Index Terms*—UML modeling, Software Engineering Education, Large Language Models, Requirements Engineering**

I. Introduction

In recent years, Large Language Models (LLMs) have demonstrated significant potential in supporting a wide range of software engineering tasks, including requirements specification, source code generation and analysis, and automated testing, with benefits deriving from the agents' inherent advanced natural language understanding and reasoning ability. The integration of LLMs into tasks related to Unified Modeling Language (UML) class diagrams such as the generation and the evaluation has emerged as a promising, although still in the early stages, example of LLM use. UML class diagrams are a fundamental notion in object-oriented designs, being particularly effective in representing domain concepts, their characteristics, and their relationship, making it a staple of university-level software engineering education. However, learning to produce correct and complete models from textual descriptions can sometimes prove to be a challenging activity for students and novice designers.

Existing work in the literature has explored the feasibility of using LLMs for the automated generation of diagrams, highlighting some degree of capability in creating reasonably complete model, although not fully close to human-made diagrams in terms on quality. Building from the existing research, this paper aims to investigate the capability of different LLM architectures in this domain through a comparison of how proprietary and open-source models are able to handle the generation of university-level modeling exercises.

For this purpose, we adopted a structured, role-based few-shot prompting strategy, aiming to let the models simulate the behavior of an expert UML modeling professor. We then analyzed the generated diagrams according to existing evaluation frameworks in literature focusing on syntactic correctness, semantic accuracy, and completeness with respect to the domain.

This article is presented as a continuation of our previous work, where we studied how OpenAI's ChatGPT model was able to generate class diagrams [1]. In regards to the original work, the following changes have been introduced:

- More LLM architectures (namely, Google's Gemini 2.5 Pro, Alibaba's Qwen 3, and DeepSeek AI's DeepDeek 3) have been introduced for a comparison of their different output. The version of ChatGPT used for this study has also been updated from the original article's GPT4 to the more recent version (at the time of performing the study, June 2025) GPT4-o.
- To improve generalizability, a different set of exercises has been collected, selecting UML class diagram modeling exercises from various Master's degree courses in Software Engineering, Computer Engineering, and Computer Science offered by Italian universities.
- The prompting strategy used during the study has been improved compared to the original work, leading to a more detailed role-based, few-shot, structured prompt.
- The focus of our analysis has been moved from comparing the diagrams generated by an LLM and those drawn by a human to comparing how different models handle the same set of requirements.
- The notation used to represent the class diagrams has been changed from PlantUML to Apollon, an open-source

UML web modeling editor that uses the JSON format to represent diagrams. This change was due to the ease of use of the latter platform, in terms of human actions that can be made on diagrams, and to allow for easier diagram comparisons.

II. Background

The application of LLMs in model-driven engineering, particularly in regards to UML class diagram creation, is a field that has been explored by different studies in recent years.

Arulmohan et al. [2] proposed a preliminary example of how LLMs could be used for the automatic generation of class diagrams, showing that it outperformed state-of-the-art natural language processing tools in terms of correct generation. Cámara et al. [3] analyzed the preliminary capabilities of ChatGPT to create diagrams, suggesting the importance of integrating the two domains (LLMs and modeling). A comparison between LLMs was performed by Chen et al. [5], highlighting that, although models displayed good domain understanding, they still struggled in terms of fully capturing the domain and following best practices for modeling. Li et al. [6] explored how to use Chain-of-Thought prompting to generate diagrams starting from user stories, showing good capabilities in the generation of classes compared to traditional prompting, although there were some issues with regards to attribute identification. Lastly, Rouabhia and Hadjadj [7] present a comparison of how different LLMs can parse use case descriptions to enhance static class diagrams by adding required methods.

The existing studies all highlight the ability to use intelligent agents to produce correct models, although with some limitations that suggest it would be better to use them as complementary tools rather than the only solution [8], [9]. To the best of our knowledge, no comparison has yet been made on how different LLM agents produce class diagrams, seeing as the current literature only discusses GPT models.

III. Study Design

We performed a comparative study to analyze the ability of different Large Language Models to generate UML class diagrams starting from textual requirements. The goal of this study, which we defined by following the Goal-Question-Metric (GQM) template [4], can be summarized as *Analyze the use of different LLM agents for the purpose of comparing UML class diagrams generated by different models starting from the same textual requirements, from the point of view of Software Modeling educators*. A summary of the study's design and goal is presented in Table I.

TABLE I
GQM template for the study

Object of study	Usage of different LLM agents
Purpose	Comparison of prompt output
Focus	Completeness and correctness
Context	Generation of UML class diagrams
Stakeholders	Software modeling educators

A. Research Questions and Metrics

The approach we adopted to evaluate the quality of the generated UML diagrams was inspired by the theoretical framework defined by Lindland et al. [10]: the framework identifies three main dimensions to assess the quality of a UML class diagram (syntactic quality, semantic quality, pragmatic quality). More precisely, we opted to focus on the syntactic and semantic quality of the generated diagrams.

The research questions we defined for the study are as follows:

- **RQ1:** Does the use of different Large Language Models for the automated generation of UML class diagram lead to differences in the quality of such diagrams?
- **RQ2:** What are the different error categories produced by different Large Language Models?

To measure the quality of the generated diagrams, we defined three metrics: the number of *Syntax Errors*, the number of *Semantic Errors*, and the *Diagram Completeness*. These metrics have been computed, for each generated diagram, according to the following procedure: I) We compared each diagram with the original solution to identify generated elements (classes, attributes, methods, relationships) that could be considered matches for elements in the reference (if their names were similar enough to represent the same concept). We then computed the *Diagram Completeness* as the count of all matching elements over the total count of elements in the original diagram; II) The diagram was analyzed a second time to identify all existing *Syntax Errors* such as using collections as attribute types, having classes not connected to the rest of the diagram, missing attribute types or multiplicity values, or using other classes as attribute names/types; III) The diagram was compared again with the reference solution to identify *Semantic Errors* in the matching elements. Errors included methods having wrong parameters or return types, associations having the wrong multiplicity or type, or attributes having the wrong type. This evaluation procedure was performed manually by one author of this paper; once the evaluation was completed, two other authors independently reviewed the results and gave their feedback. The three authors then combined all such feedback and agreed upon the final evaluation results.

B. Exercise Selection

The set of exercises used for the study was created by selecting real university-level exercises from official teaching material made available by different university courses in Computer Engineering and Computer Science of different Italian universities such as Università degli Studi di Milano, Università di Trieste, and Politecnico di Torino. The selected exercises all required the creation of a UML class diagram starting from natural-language requirements (in the Italian language) describing scenarios for different domains such as administrative, managerial, sanitary, educational, technical,

and general-purpose applications. The goal of the selection process was to have enough variation in the difficulty of the exercises: more than half of the descriptions presented some kind of ambiguity or implicit elements, at least one third of the exercises introduced explicit or implicit hierarchies, and around two thirds of the exercises presented composition and aggregation relationships.

C. Models and Prompting Strategy

To perform our comparative study, we opted to select four different Large Language Model architectures, using both cutting-edge proprietary models and emerging, but effective, open-source models, to assess the ability of different models to handle the generation of UML class diagrams. For all models, we used the freely available website solutions rather than using an API-based approach.

The four models we selected are:

- OpenAI's ChatGPT (GPT-4o). One of the best available commercial model, with good performance on different tasks and support for structured formats.
- DeepSeek AI's DeepSeek v3. A recently emerging, production-oriented open-source model optimized for code and formalized output.
- Google's Gemini 2.5 Pro. A competitive model in terms of contextual understanding and structured generation.
- Alibaba's Qwen 3. A multilingual model with good reasoning and structuring capabilities, designed for adapting to different kinds of tasks.

The prompting strategy we devised for the study was to use a role-based few-shot prompt with structured examples of how the output was supposed to be generated. We specified in the beginning of the prompt that the agent was going to act as an expert in UML modeling, to guide its behavior towards a coherent output based on simulated domain knowledge. Furthermore, each prompt contained examples of both input (in the form of the textual requirements) and output (JSON format representing a diagram), to help the model understand the necessary structure and improve the quality of the produced output. The prompt, which was used without any change for each of the four LLMs, was originally defined in the Italian language; an English translation is presented in an online appendix, together with the generated diagrams, the text of the 15 exercises (in the Italian language), and the results of our analysis[1].

D. Analysis Method

To answer RQ1, we decided to apply a fixed effect linear model on the metrics obtained with our human evaluation; more precisely, with our goal being a comparison of the performance achieved by the four different models, we performed comparisons between pairs of models considering the type of model as independent variable and the completeness and the count of syntax and semantic errors as the dependent variables. To answer RQ2, we grouped the errors performed by the four different LLMs according to their type in order to reach the total sum of errors for each category by model and we evaluated their distribution.

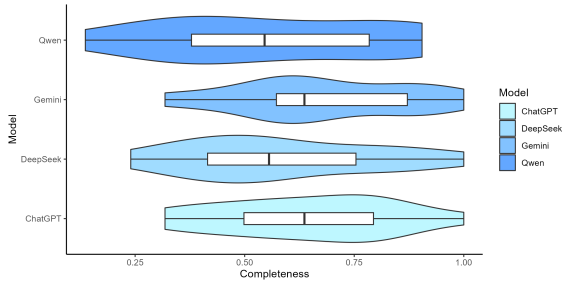
[1]https://doi.org/10.6084/m9.figshare.29492624

IV. Results

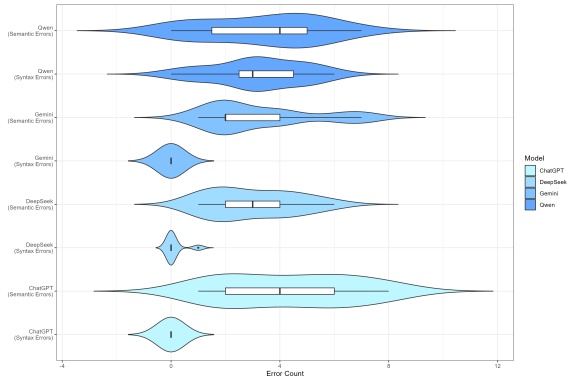
A. RQ1 - Quality of the diagrams

We present in Table II the summary statistics (minimum, maximum, mean, and standard deviation) for the three metrics of our study, for each of the four LLMs used; the graphical distribution of the three metrics is shown in Figure 1.

Fig. 1. Distribution of the three quality metrics



(a) Distribution of Diagram Completeness per model



(b) Distribution of Syntax and Semantic errors per model

For what concerns syntax errors, we can observe a good performance from ChatGPT and Gemini, as both models achieved 0 syntactic violations in all of their 15 generated diagrams; DeepSeek also performed well, producing only one syntax error in two out of 15 exercises; lastly, Qwen is the model with the worst performance, with only one solution out of the 15 exercises having 0 syntactic errors, and an average of around 3 errors per exercise.

Regarding semantic errors, the model with the overall best performance is DeepSeek, with the lowest maximum number of errors and the lowest mean value (3 errors) as well; Qwen and Gemini perform similarly well, producing only slightly more errors on average compared to DeepSeek; ChatGPT is the worst performing model regarding semantic errors, implying that, while good at representing the necessary concepts, it may not be the best model at conveying additional domain constraints (e.g. association multiplicity, attribute typing).

Lastly, the results obtained for the Completeness metric show that Gemini and ChatGPT are the best performing

TABLE II
MINIMUM, MAXIMUM, AND MEAN VALUE (WITH STANDARD DEVIATION) OF THE THREE METRICS, GROUPED BY MODEL

	ChatGPT			DeepSeek			Gemini			Qwen		
	Min	Max	Mean (SD)	Min	Max	Mean (SD)	Min	Max	Mean (SD)	Min	Max	Mean (SD)
Syntax Errors	0	0	0 (0)	0	1	0.13 (0.35)	0	0	0	0	6	3.2 (1.7)
Semantic Errors	1	8	4.33 (2.44)	1	6	3 (1.60)	1	7	3.27 (2.02)	0	7	3.4 (2.20)
Completeness	31.8%	100%	64.2% (0.20)	24%	100%	58.3% (0.22)	31.8%	100%	67.7% (0.20)	13.6%	90.5%	55.4% (0.25)

models, with somewhat similar scores (equal min and max, and 65% avg. completeness for both); DeepSeek and Qwen perform slightly worse, reaching around 55% completeness on average, implying that both models are still capable to represent most of the required concepts; the latter, however, never managed to produce a diagram containing all required elements (100% completeness), implying that it may not be the most effective for complete domain representation.

Table III contains the results of our fixed effects linear model applied between each pair of models; for each pair, we present the estimate (considering the second model in the pair against the first), and the corresponding p-value.

TABLE III
RESULTS OF THE LINEAR MODEL APPLIED BETWEEN LLM PAIRS. STATISTICALLY SIGNIFICANT P-VALUES ARE IN **BOLD**

Model Pair	**Syntax Errors**	**Semantic Errors**	**Completeness**
ChatGPT - DeepSeek	0.133 (0.675)	-1.33 (**0.086**)	-0.06% (0.462)
ChatGPT - Gemini	3.932e-15 (1)	-1.067 (0.167)	0.04% (0.661)
ChatGPT - Qwen	3.2 (**3.15e-14**)	-0.933 (0.226)	-0.9% (0.278)
DeepSeek - Gemini	-0.133 (0.675)	0.267 (0.728)	0.09% (0.462)
DeepSeel - Qwen	3.067 (**1.45e-13**)	0.400 (0.602)	-0.03% (0.724)
Gemini - Qwen	3.2 (**3.15e-14**)	0.133 (0.862)	-0.12% (0.130)

The linear model shows that most comparisons between models do not yield statistically significant differences in performance: the only relevant difference is in the performance obtained by Qwen for syntax errors. There is, however, a slightly significant difference in the number of semantic errors when comparing ChatGPT with DeepSeek, with the latter having a lower number, on average.

Answer to RQ1: Each of the four LLMs show different capabilities and limitations. ChatGPT, DeepSeek, and Gemini all show excellent performance concerning UML syntax rules, producing either none or very few errors, while Qwen appears to struggle in this regard. DeepSeek showed the best performance in terms of semantic correctness, implying a good ability to understand and represent domain constraints; Qwen and Gemini achieved similarly good results, while ChatGPT had the worst overall metrics for semantic errors. All models obtained above average completeness scores, indicating good ability to solve modeling exercises, with the proprietary models (ChatGPT and Gemini) performing better than open source ones.

B. RQ2 - Error categories

To answer RQ2, we counted the number of both syntax and semantic errors found during the evaluation of the diagrams produced by the four models, and computed the total sum of errors for each category.

The distribution of syntax errors is presented in Figure 2(a): in line with the analysis presented earlier in this section, we see that almost all syntax errors have been found in the diagrams generated by Qwen. More precisely, most of the errors consisted of using collections rather than associations to express one-to-many and many-to-many relationships, and including attributes that, either with their name or their type, acted as foreign keys that referenced other classes, thus violating some of the prompt's constraints.

Regarding semantic errors, whose distribution is shown in Figure 2(b), we observe a more varied distribution of errors. The most obvious weakness of the four models is the incorrect representation of associations: most of the errors come from using the wrong type for an association (e.g. composition in place of aggregation, a regular association rather than a generalization, or a special relationship where a regular one was expected), or from using incorrect multiplicity values.

Interestingly enough, there are two different models with the highest number of errors in the most common categories: ChatGPT obtained most of the errors for association typing (almost twice more than the second place model for the category), while Qwen struggled the most with understanding multiplicities, although with a less pronounced difference compared to the other models.

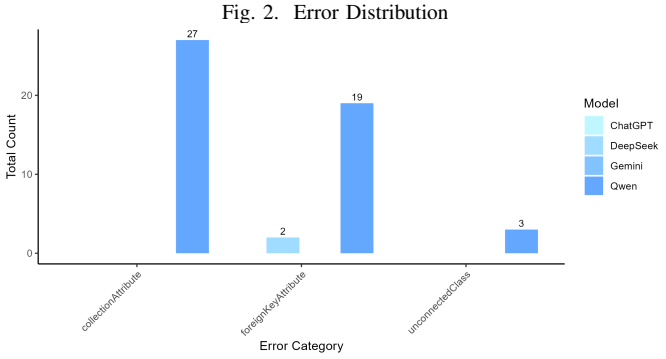
Other error categories appear to be far less distributed, and with less pronounced differences in performance: all four models struggled more or less the same when it came to representing methods as attributes instead, understanding the order of classes in aggregation/composition relationships, or assigning the correct type to attributes and methods.

Answer to RQ2: Syntax errors were something that only Qwen struggled with, since the other three models showed a better ability to adhere to UML modeling rules. Regarding semantic errors, all four models had trouble in representing associations coherently with the domain specifications; selecting the correct association type or multiplicity values proved to be the most common errors for the models. Other violations, such as inverting the order of classes in some relationships, assigning wrong types to attributes or methods, or representing methods as attributes, were common to all four models, but with much lower frequency.

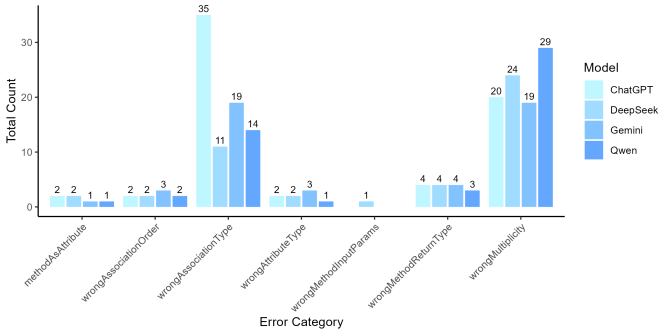
V. THREATS TO VALIDITY

This section presents, in accordance to the four categories defined by Wohlin et al. [11], the possible threats to the validity of our study.

Fig. 2. Error Distribution



(a) Distribution of syntax errors per model



(b) Distribution of semantic errors per model

Threats to internal validity: although we tried to mitigate the bias in the single evaluation by having its results assessed by two additional authors, it may be possible for other evaluators to achieve different results. The prompt is also a possible cause of concern: there is no guarantee that the prompting strategy was the most effective for our goal, using examples may have led to overfitting, specific parts of the prompt may have guided the output in a suboptimal way; lastly, different prompts may have worked better for different models rather than a single one.

Threats to construct validity: the metrics we adopted may not be the most significant in assessing the quality of the generated diagrams. Regarding errors, other error categories could have been used in addition to the ones we defined, while diagram completeness could have been computed in a different way compared to a simple ratio of correctly identified elements.

Threats to external validity: we cannot state for certain that the results obtained with our prompt could be generalized and applicable to different UML class diagram formats, as well as different modeling notations.

VI. Conclusion

In this paper, we performed a comparison between four different Large Language Model agents to assess their ability to generate UML class diagrams, with respect to the overall diagram quality and number of errors. For this purpose, we evaluated the generated diagrams by hand, performed a statistical analysis of the metrics obtained in this way, and categorized the different errors we encountered.

The results of the statistical analysis yielded only one significant result, that is, the lower performance obtained by Qwen in terms of following UML syntax conventions, something that the three other models managed to handle with no issues. Our interpretation of these results is that all four models are more or less equally capable to generate class diagrams with good quality, with each model presenting different strengths and weaknesses (e.g. DeepSeek had the best performance for semantic errors, showing good ability to follow domain specifications, while ChatGPT and Gemini obtained better results in terms of overall correctness).

Seeing as this work is a continuation of a previous article where we performed a single comparison between human-made and GPT-generated diagrams, we intend to continue exploring this research area: our next goal is to implement a tool that, through dedicated APIs, is able to assist modeling educators by providing them with automatic diagram generation, which should prove to be effective in designing new solutions.

References

[1] De Bari, D., Garaccione, G., Coppola, R., Torchiano, M. & Ardito, L. Evaluating Large Language Models in Exercises of UML Class Diagram Modeling. *Proceedings Of The 18th ACM/IEEE International Symposium On Empirical Software Engineering And Measurement*. pp. 393-399 (2024), DOI: https://dl.acm.org/doi/10.1145/3674805.3690741

[2] Arulmohan, S., Meurs, M. & Mosser, S. Extracting Domain Models from Textual Requirements in the Era of Large Language Models. *2023 ACM/IEEE International Conference On Model Driven Engineering Languages And Systems Companion (MODELS-C)*. pp. 580-587 (2023)

[3] Cámara, J., Troya, J., Burgueño, L. & Vallecillo, A. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software And Systems Modeling*. **22**, 781-793 (2023,6,1), https://doi.org/10.1007/s10270-023-01105-5

[4] Basili, Victor R. "Software modeling and measurement: the Goal/Question/Metric paradigm." 1 Sep. 1992

[5] Chen, K., Yang, Y., Chen, B., Hernández López, J., Mussbacher, G. & Varró, D. Automated Domain Modeling with Large Language Models: A Comparative Study. *2023 ACM/IEEE 26th International Conference On Model Driven Engineering Languages And Systems (MODELS)*. pp. 162-172 (2023)

[6] Li, Y., Keung, J., Ma, X., Chong, C., Zhang, J. & Liao, Y. LLM-Based Class Diagram Derivation from User Stories with Chain-of-Thought Promptings. *2024 IEEE 48th Annual Computers, Software, And Applications Conference (COMPSAC)*. pp. 45-50 (2024)

[7] D. Rouabhia and I. Hadjadj, Behavioral Augmentation of UML Class Diagrams: An Empirical Study of Large Language Models for Method Generation, *arXiv preprint* arXiv:2506.00788, 2025. Available: https://arxiv.org/abs/2506.00788.

[8] Shehata, M., Lepore, B., Cummings, H. & Parra, E. Creating UML Class Diagrams with General-Purpose LLMs. *2024 IEEE Working Conference On Software Visualization (VISSOFT)*. pp. 157-158 (2024,10), DOI: https://ieeexplore.ieee.org/abstract/document/10794946,ISSN:2832-6555

[9] Wang, B., Wang, C., Liang, P., Li, B. & Zeng, C. How LLMs Aid in UML Modeling: An Exploratory Study with Novice Analysts. *2024 IEEE International Conference On Software Services Engineering (SSE)*. pp. 249-257 (2024,7), DOI: https://ieeexplore.ieee.org/abstract/document/10664407/references

[10] Lindland, O., Sindre, G. & Solvberg, A. Understanding quality in conceptual modeling. IEEE Software. **11**, 42-49 (1994) DOI: https://doi.org/10.1109/52.268955

[11] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B. & Wesslén, A. Experimentation in software engineering. (Springer Science & Business Media,2012) DOI: https://doi.org/10.1007/978-3-642-29044-2