Augmenting, Not Replacing: The Role of Generative AI in Teaching Software Modeling

Maria Vitória Costa do Nascimento
Instituto de Computação,
Universidade Federal do Amazonas
Manaus, Brazil
vitoria.nascimento@icomp.ufam.edu.br

Márcia Sampaio Lima
Universidade do Estado do Amazonas
Manaus, Brazil
msllima@uea.edu.br

Tayana Uchôa Conte
Instituto de Computação,
Universidade Federal do Amazonas
Manaus, Brazil
tayana@icomp.ufam.edu.br

ABSTRACT

Context: Software Modeling is a fundamental component of Computer Science curricula, addressing diverse themes including domain modeling. Despite its importance and years of research, a persistent challenge remains: students' difficulty in comprehending and applying core modeling concepts. While numerous studies have identified these issues and proposed solutions, many involving collaborative and feedback-based environments, the recent advances of Large Language Models (LLMs) offer a new frontier. This study proposes and evaluates a pedagogical approach that leverages a generative LLM to improve students' practical skills in modeling UML class diagrams. **Method:** In an empirical study with 30 undergraduate students, participants first created a class diagram for a given problem, then used an LLM for the same task, and finally compared both models. We analyzed quantitative pre- and post-study self-assessments and qualitative feedback on the process. **Results:** The study revealed a statistically significant improvement ($p<0.05$) in students' self-assessed knowledge for class diagrams, an effect not observed in control topics. Qualitatively, students praised the LLM for accelerating ideation and identifying errors, but also noted significant inaccuracies in the generated models, particularly in class relationships. Students perceive the LLM not as an autonomous 'autopilot' for generating solutions, but as a valuable yet flawed 'co-pilot' that augments the learning process. This suggests LLMs are effective tools for support and validation rather than direct solution generation.

KEYWORDS

Software Engineering Education, Large Language Models, Artificial Intelligence in Education, UML Class Diagrams, Empirical Study

1 Introduction

The process of teaching and learning Unified Modeling Language (UML) diagrams, particularly class diagrams, presents significant challenges [18]. Many students struggle with the abstract nature of modeling, which is often taught using traditional, instructor-centered methods that focus heavily on theoretical concepts [5]. Cardoso et al. [4] and Sien et al. [18] have identified and categorized various difficulties students face when constructing UML diagrams. The main problems include the overall construction of the diagrams, understanding the objects and the reading flow of each diagram, definition of classes, messages and loops in sequence diagrams, generalization, and inheritance.

Collaborative learning through team-based diagram creation has been proposed as a solution by Sien et al. [18], enabling shared learning and collective problem-solving, however it has its drawbacks. Students frequently report concerns regarding group dynamics, citing issues like difficulty maintaining focus, tight deadlines, inequitable workload distribution, and amplified confusion when collective understanding is lacking [20].

In this context, Large Language Models present a compelling and innovative alternative for fostering interactive and collaborative student engagement. Functioning as on-demand, individualized partners, LLMs can serve as virtual tutors that facilitate a one-on-one learning environment [8]. Their ability to provide instant, detailed explanations for their generated outputs introduces a powerful pedagogical dynamic [22], allowing students to not only see an alternative solution but also to understand the reasoning behind it.

Recent studies [3] have showed that LLM has been used in all different areas of SE such as: Requirements Engineering, Systems Design and Code Generation. Ferrari et al. [7] and Camara et al. [6] have used LLM to generated class and sequence diagrams, both studies show that LLMs have the capability of generating UML diagrams, however it has its issues specially with weak requirements or very specific context.

To address this gap, the study has three primary objectives. First, we aim to quantitatively measure the impact of a novel LLM-based pedagogical methodology on students' self-perceived knowledge of UML class diagrams. Second, we seek to qualitatively explore the student experience, identifying the perceived benefits and limitations of using an LLM as a modeling collaborator. Finally, by synthesizing these quantitative and qualitative findings, we aim to characterize the emergent role of the LLM in the software design learning process, moving beyond a simple evaluation of its technical accuracy.

To achieve the proposed goals, we conducted an empirical study with a group of undergraduate students enrolled in the Systems Analysis and Design course in the Computer Science program. We applied a methodology that integrated the use of LLMs and PlantUML to construct UML class diagrams. The students engaged in comparative exercises, analyzing diagrams produced independently against those generated with LLM assistance. For the generation with the LLM, all students used the same prompt and same scenario in the same model (Free ChatGPT[1]). We collected self-assessment

[1] https://chatgpt.com/

data from the students before and after the study, along with qualitative information about the challenges they faced and the impact of the study on their learning process. Analysis of the data revealed a statistically significant improvement in students' self-assessed knowledge in class diagrams and showed that students view LLM as a support for their modeling process, while still recognizing some issues of the tool.

The main contributions of this work are: (1) the design and empirical evaluation of a novel pedagogical methodology centered on the critical comparison between student-generated and LLM-generated UML diagrams; (2) a deep qualitative analysis of students perceptions, which characterizes the nuanced, dual role of the LLM. Identifying its function as a augmentation tool rather than replacing their knowledge in modeling.

The remainder of this paper is structured as follows: Section 2 introduces the fundamental concepts of Large Language Models, along with a review of related works. Section 3 details our research methodology. Section 4 presents the quantitative and qualitative findings of our study. Section 5 distills the key lessons learned from this experience, while Section 6 acknowledges the study's limitations. Finally, Section 7 concludes the paper and outlines potential avenues for future work.

2 Background

This section provides an overview of the foundational concepts of Large Language Models (see Subsection 2.1) and PlantUML (see Subsection 2.2) together with the Related Work (see Subsection 2.3).

2.1 Large Language Models

Large Language models are artificial intelligent models based on the transformer architecture [21], trained in vast textual databases to understand and generate text. These Large Language Models, different than other Language Models are capable of developing emergent abilities, where a large model can execute tasks that wasn't trained on by prompting paradigms, such as few-shot and chain of thought [23]. It is this advanced capacity for contextual reasoning and generation that makes them powerful tools for domains like education.

The use of large language models in education has been identified as a potential area of interest due to the diverse range of applications they offer [10]. This trend is quantified by a recent survey in Computer Science education, which found that LLMs are most frequently used with undergraduate students, primarily in "Introduction to Programming" courses, with Software Engineering being the fourth most-researched subject [16]. The survey also highlights a positive student perception, with students valuing the models' ability to provide explanations and viewing them as supplementary teaching assistants.

In our study, we explore the use of LLMs to explain the process of designing a UML class diagram and create a diagram for a context so that students can compare alternatives and critically analyse both results.

2.2 PlantUML

PlantUML is a open-source textual UML tool which allows developers to create a wide range of UML diagrams using a simple and intuitive syntax [15]

The choice of a text-to-diagram approach was a crucial technical prerequisite for this study. Large Language Models are fundamentally text-processing engines. Therefore, providing them with a structured, textual representation of a diagram is more reliable than asking them to interpret a graphical image. This methodology has become a standard in recent literature exploring LLM-based diagram generation [6, 7]. We selected PlantUML specifically due to its extensive adoption, robust feature set, and—most importantly—the high fluency that current LLMs demonstrate with its syntax, likely due to its prevalence in public code repositories and documentation used in their training data [14].

2.3 Related Works

Studies [4, 5, 18] have showed students challenges when learning UML diagrams. In response, the research community has explored two primary avenues to enhance UML education: pedagogical strategies and specialized software tools.

For instance, Silva et al. [19] investigated the impact of five different active learning strategies on learning UML. Through focus group sessions and post-modeling questionnaires, they found high student agreement levels for strategies like Inspection-Based learning (91.7%) and Think-Pair-Square (81.1%). A key finding was that these strategies fostered collaboration and that students particularly valued the immediate feedback inherent in the Inspection-Based approach.

The use of examples as a learning aid is another prominent strategy. Karasneh et al. [9] demonstrated that students who used a repository of existing models created diagrams that were 18% better than those of a control group. Furthermore, students who initially created models without examples improved their work by 19% after being given access to the repository, also reporting higher confidence in their final designs.

These studies highlight the importance of having a collaborative and feedback full environment for the students. Some research has tried to adapt these needs within a tool-based research, developing various software tools. Alhazmi et al. [1] created a tool to teach Sequence Diagrams that provides instant feedback as students model. Over 80% of student participants reported that this real-time feedback improved their understanding.

Similarly, COLLECT-UML is an Intelligent Tutoring System (ITS) designed to teach object-oriented design using class diagrams [2]. The system guides students by providing constraint-based feedback on their diagrams. An evaluation showed that students achieved significantly higher scores on a post-test after using the system and felt that more time with the tool would lead to greater learning.

These studies with traditional tools and Intelligent Tutoring Systems (ITS) highlight a clear trend: effective, real-time feedback is crucial for improving students' UML modeling skills. However, these systems often face a significant limitation: their feedback mechanisms are based on pre-programmed rules and constraints. While effective for detecting specific syntactical errors, they lack the flexibility to understand the student's design intent, provide

deeper semantic feedback, or explain complex design principles in natural language.

To address these limitations, the use of LLMs in Software Engineering education has risen significantly. A systematic review by Khan et al. [11] on their use in Finnish institutes identified four key application themes: enhancing learning experiences through personalization, improving assessment and feedback, fostering collaboration with AI learning partners, and accelerating skill development in tasks like programming and debugging. Beyond structured educational settings, observational studies show how software engineers organically use these tools in their daily work. Khojan et al. [12] found that, aside from code generation, one of the most common use cases for ChatGPT was for learning and receiving high-level guidance. While participants found the tool very useful for learning (75% agreement), they also stressed the need to critically verify its suggestions. This highlights a key dynamic: LLMs are perceived as powerful partners for guidance and learning, but require human oversight.

Building on this role as a tool for high-level guidance, researchers have begun to specifically explore the capability of LLMs in the abstract task of diagram generation.

Ferrari et al. [7] explored ChatGPT's ability to generate UML sequence diagrams from natural language requirements. Their results showed that the generated diagrams scored well for understandability and standard compliance, but exhibited issues with completeness and correctness, especially when requirements were of low quality. Similarly, Camara et al. [6] investigated ChatGPT's use for constructing class diagrams with OCL constraints. The study concluded that while ChatGPT generally produces syntactically correct diagrams, its semantic correctness is less reliable than its code generation capabilities, and the problem domain has a remarkable impact on the quality of the results.

A key technical consideration for applying LLMs to UML modeling is the translation from visual diagrams to a format the AI can process. Ferrari et al. [7] and Camara et al. [6] overcame this by adopting a text-to-diagram approach (e.g., using PlantUML). This represents diagrams as structured code, a format native to LLMs, which allows for the precise analysis and generation required by our interactive learning methodology.

In summary, the reviewed literature indicates that while LLMs can successfully generate syntactically correct UML diagrams, they consistently struggle with semantic correctness and completeness [6, 7]. The current research has largely focused on evaluating the technical quality of these AI-generated artifacts. A gap remains, however, in understanding how the process of student interaction with these imperfect models can be structured for pedagogical benefit.

This study addresses this gap by proposing and evaluating a novel learning methodology centered on comparative analysis. Instead of simply using the LLM to obtain a correct answer, our approach requires students to first create their own solution and then critically compare it with an LLM-generated alternative. We aim to understand how this reflective practice influences their perceived knowledge, their ability to identify design trade-offs, and their overall perception of LLMs as learning tools.

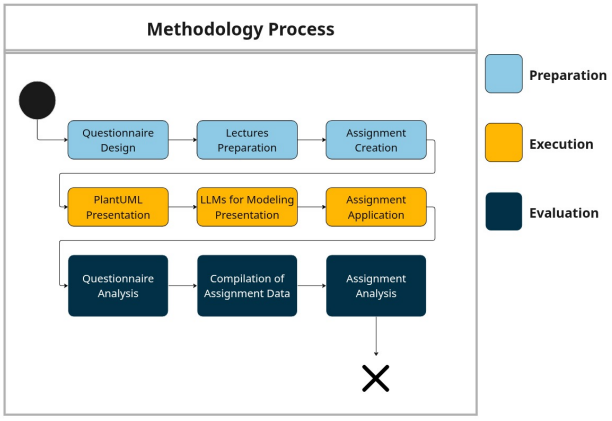

Figure 1: Methodology Process

3 Methodology

The study involved 31 undergraduate students from a 6th-semester "Systems Analysis and Design" course in a Computer Science program. With "Introduction to Software Engineering" as a prerequisite. The course focuses to teach modern technologies for the modeling, analysis, and design of computer systems.

The methodology process is shown in Figure 1. In the Preparation Phase (see Subsection 3.1) we will describe the creation of study artifacts. In the Execution Phase (see Subsection 3.2), we will detail the execution of the lectures and the assignment done by the participants. In the Evaluation Phase (see Subsection 3.3) we will explain the data analysis procedures.

3.1 Preparation Phase

This subsection describes the creation of the artifacts (See section 7) used for the study's execution. This phase consisted of three main steps, as shown in the "Preparation" block of Figure 1.

Questionnaire Design: To measure the changes in students' perceived knowledge, a data collection strategy was implemented using pre- and post-study questionnaires. The pre-study questionnaire collected a self-assessment of the students' knowledge, in a continuous score from 0.0 (No Knowledge) to 5.0 (Advanced Knowledge), for class, sequence, state, and activity diagrams. It also included an open-ended question about their biggest difficulty in modeling. The post-study questionnaire repeated the self-assessment items to track changes and included new qualitative questions. These questions asked students if their scores changed, how they would rate between 0.0 and 5.0 the experience of using an LLM as a modeling aid, if the use of the LLM impacted their grades, and whether they would use an LLM for this purpose again.

Lectures Preparation: Given that textual UML, specifically PlantUML, was a novel framework for the participants, we developed dedicated instructional materials. The objective was to equip students with the skills needed to understand and critically evaluate diagrams generated by the LLM. A lecture was designed to introduce PlantUML fundamentals, supported by a slide presentation and a "cheat sheet" summarizing key syntax. A second lecture, "Using LLMs for Modeling," formally introduced the research context

and the final assignment. For this assignment, a specific prompt was engineered to guide the LLM's output. The prompt was: *"Crie um diagrama de classe, usando plantUML, para o contexto dado, e explique os passos que foram feitos para atingir o resultado. Input: "*. This prompt was intentionally designed to be straightforward, compelling the LLM not only to generate the diagram but also to articulate its creation process, a crucial component for enabling student analysis.

Assignment Creation: A custom modeling scenario was developed for the final assignment. The scenario was scoped to focus on core UML concepts relevant to the students' learning level, specifically class identification, inheritance, relationships, and multiplicity. A ground-truth diagram was also created by the researchers to serve as a baseline for internal evaluation purposes.

3.2 Execution Phase

This subsection details the in-class activities conducted during the study. The execution phase, as illustrated in the execution group of Figure 1 the process had three main steps, it was conducted over two 120-minutes class sessions, held one week apart. The pre-study questionnaire was administered at the beginning of the first session, and the post-study questionnaire was administered at the end of the second session, after the completion of all tasks. The activities within these sessions are detailed below.

PlantUML Lecture: The first session began with a lecture introducing the rationale behind text-based UML and the specific syntax of PlantUML. To ensure practical understanding, the session included hands-on exercises focused on creating class diagrams. The previously prepared ´´cheat sheet" served as a quick reference guide for students during this activity.

LLMs for Modeling Lecture: The second session started with a lecture that formally introduced the research context, explaining why and how LLMs would be used as a learning aid. To prepare students for the main assignment, a brief demonstration was conducted where a sample scenario was solved using the engineered prompt. Students were guided to analyze both the diagram generated by the LLM and its explanatory steps, serving as an on boarding process for the main task.

Assignment Application: Following the LLM lecture, the remainder 90-minutes was dedicated to the core data collection activity. The assignment required students to progress through three stages:

- **Independent Creation:** Students were first instructed to individually model a class diagram based on the provided scenario, using the PlantUML skills they had acquired.
- **LLM Generation:** Subsequently, they used the provided prompt to have the LLM generate a class diagram and an explanation for the same scenario.
- **Comparative Analysis:** The final step required students to write a comparative analysis. They were prompted to reflect on the differences between their diagram and the LLM's, identify any new insights gained, and articulate their overall thoughts on the process.

3.3 Evaluation Phase

This subsection explains the data analysis procedures. As detailed in the Evaluation group of the Figure 1, the process consisted of three steps:

Questionnaire Analysis: Data from the pre- and post-study questionnaires were analyzed both quantitatively and qualitatively. The quantitative analysis focused on the self-assessment scores (detailed in Subsection 4.1), involving the generation of descriptive statistics and the execution of a statistical hypothesis test. The qualitative analysis of the open-ended questions involved generating word clouds and identifying common themes in student responses (detailed in Subsection 4.2).

Assignment Compilation: The artifacts from the in-class assignment were systematically compiled. To facilitate analysis, all handwritten comparisons were transcribed into a spreadsheet, linking each reflection to the corresponding participant and their diagrams.

Assignment Analysis: The transcribed comparisons from the assignment were analyzed qualitatively (see Subsection 4.2). We employed a thematic analysis approach to identify common patterns in students' experiences. The emergent themes were then categorized as Positive, Negative, or Reflective.

4 Results

This section presents the findings from our empirical study. The results are organized into two main parts: first, we detail the quantitative findings from the pre- and post-study questionnaires (Subsection 4.1). Second, we present the qualitative findings, which are detailed sequentially according to the data collection instrument used (Subsection 4.2)

4.1 Quantitative

The quantitative analysis was based on the data from pre- and post-study questionnaires. In both, students self-assessed their knowledge on a continuous scale from 0 to 5 for four UML diagrams: Class, State, Activity, and Sequence. We included all four diagram types in the pre-study questionnaire to avoid introducing response bias and to establish a baseline. This design allows us to isolate the intervention's effect on Class Diagram knowledge from any general learning effects that might occur across all topics.

As shown in Table 1, the pre-study results indicate that the median self-assessed knowledge was similar across all diagrams, centering around a score of 3.0. However, the means and standard deviations for Activity and Sequence diagrams were lower and higher, respectively, suggesting more variability and less confidence in these areas compared to Class and State diagrams.

Table 1: Descriptive Statistics - Pre-Study Scores

	Class	State	Activity	Sequence
Median	3.000	3.750	3.000	3.000
Mean	3.464	3.554	3.193	2.768
Std. Deviation	0.706	0.916	1.006	1.014
Range	3.000	4.000	4.000	4.000

Following the intervention, the post-study results (Table 2) show a marked improvement in the self-assessed knowledge for Class Diagrams. The median score increased to 4.0. Notably, the standard deviation and range for Class Diagrams became the lowest among all diagram types, suggesting that the intervention not only increased students' perceived knowledge but also led to a more uniform level of confidence across the cohort.

Table 2: Descriptive Statistics - Post-Study Scores

	Class	State	Activity	Sequence
Median	4.000	4.000	3.000	3.000
Mean	3.845	3.638	3.352	3.072
Std. Deviation	0.656	0.718	0.701	0.808
Range	2.500	3.000	3.000	4.000

To verify if the observed improvement in Class Diagram scores was statistically significant, we first performed a Shapiro-Wilk test [17] to assess data normality. The test yielded a p-value below our significance level of a=0.05 (Figure 3), indicating that the data are not normally distributed.

Table 3: Shapiro-Wilk Normality Test Results by Diagram and Period

Diagram Type	Period	Shapiro-Wilk	P-value
Class	Pre-Study	0.860	.001
	Post-Study	0.847	< .001
State	Pre-Study	0.880	.004
	Post-Study	0.854	< .001
Activity	Pre-Study	0.876	.003
	Post-Study	0.859	.001
Sequence	Pre-Study	0.904	.014
	Post-Study	0.889	.006

Consequently, we used the non-parametric Mann-Whitney U test [13] to compare the pre- and post-study scores for Class Diagrams. The results, presented in Table 4, show a statistically significant difference (p<0.05). This indicate that the proposed learning process had a positive impact on the students' perception of their knowledge.

Table 4: Mann-Whitney U test

	U test	p
Class	284.500	.038
State	389.500	.785
Activity	388.500	.773
Sequence	319.500	.147

In addition to self-assessed knowledge, the post-study questionnaire measured the students' perception of the LLM's helpfulness

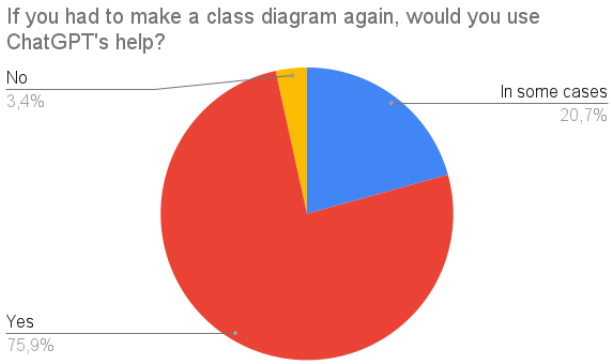

Figure 2: If you had to make a class diagram again, would you use ChatGPT's help?

on a continuous scale from 0.0 to 5.0. The median score was 4.0, with a mean of 3.74 and a standard deviation of 0.95 (Table 5). These results indicate that the majority of students perceived the LLM assistant as a helpful learning support tool.

Table 5: Descriptive Statistics - LLM Help Perception

	LLM Help Perception
Median	4.000
Mean	3.741
Std. Deviation	0.951
Range	3.000

This positive perception is further supported by students' intent for future use. When asked if they would use the LLM-based process again for a similar modeling task, 75.9% responded "Yes", and 20.7% responded "In some cases". Only a small fraction (3.4%) stated they would not use it again (Figure 2), confirming the high utility and acceptance of the proposed approach.

4.2 Qualitative

The qualitative data were gathered from three sources: a pre-study questionnaire, the final assignment, and a post-study questionnaire. To provide a clear account of the student journey, the findings from each instrument are presented in the order they were collected.

4.2.1 Pre-Study Questionnaire. In the first self-assessment questionnaire, the students were asked "What is your biggest difficulty in Modeling Diagrams?" With the responses it was generated a word cloud (Figure 3) to identify the most frequent terms in the students' answers. To ensure a more precise analysis, it was excluded from the word cloud the words *diagram* and *diagrams*, because as the study is based on UML diagrams, it is expected that these terms would be repeated in the answers.

The words in emphasis are: sequence, class, modeling, identifying, and visualizing. This result reinforces the challenges students have with Sequence and Class diagrams: identifying important parts of the scenario and the adequate visualization of the diagrams.

Figure 3: Word Cloud of Students Difficulties

4.2.2 Assignment. The data used in this subsection was collected by Question 3 of the assignment. We categorized the participants' comments into three categories: positive, negative, and reflective.

Positive: According to the participants, the positive aspects of using LLM to create class diagrams were:

- **Helping with forgotten or wrong attributes**, participants stated that the LLM diagram was helpful to notice attributes they've had forgotten. P7: *"GPT showed some attributes that I forgot, such as: Payment: I didn't include the payment_date attribute, User: I didn't include the email attribute, Material: I didn't include the title attribute"*, P9: *"The version generated by LLM was very good at demonstrating attributes and methods of some classes, something I don't think about very clearly, so it can be useful when helping to understand this part."*, P26: *"Unlike my diagram, LLM highlighted several attributes and methods of each class that are not in my diagram."*
- **Creation of better classes**, the students cited cases where the alternative generated by the LLM seemed better and more simplified than their alternative. P15: *"The Loan class was a major point of contention between our models. I handled the late payment fine by including an attribute with the fine amount in the Loan class. GPT, on the other hand, created the Fine class, which stores the fine amount and whether it was paid, in addition to adding the status attribute to loan. With this, GPT partially decoupled these entities, allowing us to easily check whether there are late payments or outstanding fines. In my approach, the fine would be calculated by a method in the User class and not stored. I recognize that GPT's approach makes more sense."*, P31: *"Regarding the Material class, I believe it was better developed by LLM, as I created two classes to represent physical and digital materials, however, LLM transformed these two classes into a single attribute (format) in Exemplar, this greatly simplified the diagram."*
- **Aid with Missing Classes** participants stated that they've forgotten classes and GPT showed those classes. P19: *"The loan class was a class that didn't cross my mind to create and it makes total sense the way the chat used it in the diagram."*, P24: *"The initial point I noticed was my mistake in forgetting the librarian class, which is extremely necessary."*
- **Assistance identifying relations** similarly to the other aspects of the class diagram, students related that ChatGPT helped them identifying errors in their relations between classes. P15: *"In the issue of paying a fine, chatGPT associated the corresponding class (payment) with the loan, while my vision was to associate it with the user, I consider that chatGPT delivered a better solution in this case."*, P4: *"The LLM version identified that the relationship between Exemplar and Material is an association."*, P12: *"The main change that the chat made was the use of inheritance, something that I didn't use"*

Negative: However, students also stated some challenges with the usage of the tool.

- **The diagram generated lacks some attributes**, some students related that the diagrams generated by the LLM had some attributes from the context missing. P21: *"Several attributes and associations are missing in a way that is far from the level of detail of the statement."*, P30: *"Finally, I realized that the version generated by chatGPT, despite being more semantically correct according to the UML, leaves something to be desired in some aspects, such as the lack of some attributes"*
- **Absence of some Classes** participants also stated that some diagrams didn't have all class stated in the context. P6: *"GPT did not create the fine and history classes, which I think are very important for the specified system."*, P1: *"LLM did not create the Class class that is specified in the text and can be created by teachers and may have materials associated with it."*
- **Issues with relations** participants cited cases where in the diagrams generated by LLM they encountered classes with no relations, double relations and forgot some relations. P16: *"LLM did not link the loan to the copy LLM did not link the researcher to the material"*, P23: *"The generated Payment class is not related to any existing one."*, P31: *"Regarding the image generated by LLM, I was able to clearly identify that the Fine class is not related to any other class, which in my view is wrong, as fines should be applied to loans that exceed the return date."*
- **Didn't generate multiplicity** students also noticed cases where the LLM didn't generate the multiplicity in the diagram. P7: *"The LLM, for some reason, did not generate cardinality of relationships and also did not generate association tables or enums."*, P12: *"The LLM did not include multiplicity."*, P6: *"Did not use multiplicity, which makes it difficult to view the diagram"*

Reflexive: Some students also presented reflexive comments, pondering about both good and bad aspects of the experience:

Figure 4: Word Cloud Positive Answers

- **Using LLM in the beginning stages**: Students pointed out that a good usage of the tool, even with its flaws, is to use the diagram generated as a first-draft. P1: *"I partially agree with what was done by LLM, but the result is certainly already minimally satisfactory to evaluate as an initial survey."*, P15: *"I found the GPT modeling to be very good, but incomplete. I would use it again as a basis for my modeling."*
- **Use LLM as a complementary tool**: Students also stated that they would use the tool again but as a complementary tool to their own knowledge. P24: *"In my opinion, the generator is a good complementary tool, that is, for adding or recommending editing details that we failed to notice or not notice in the first analysis."*, P28: *"I found it very interesting to see what the LLM version proposed. I believe that by taking all the good points from both diagrams it would be possible to reach a great result because LLM understood, I understood, and vice versa."*

4.2.3 Post-Study Questionnaire. In the post-study questionnaire, the students were asked if the scores from this assessment were different from the first assessment. 18 students answered negatively or stated they didn't remember their past scores. The answers of the other 11 students were positive and mentioned the activities and exercises done during the study and terms that indicate improvement, as *understand*, *easier*, and *better*. These terms can be seen in the word cloud presented in Figure 4.

The students also answered the question "How was your experience using ChatGPT for modeling? It helped? Did you have any limitations?" From the answers it was identified some aspects about the general use of ChatGPT in the process of modeling diagrams.

Among the positive comments cited, it highlights:

- **Assistance in defining methods and attributes,** some students stated that the tool contributed to the visualization and identification of attributes and methods in classes. Some examples of this perceptions are: P2: *"It helped me visualize attributes, which I'm terrible at."*, P7: *"gpt chat greatly accelerated the process of creating classes and attribute definitions"*, P15: *"It only helped with identifying attributes and methods of possible classes"*, P24: *"it showed me interesting ways to use methods that I liked"* and P27: *"gpt chat is good at class attributes, it doesn't forget anything"*.
- **Attention to details,** students mentioned that ChatGPT has helped them notice aspects that otherwise would gone unnoticed during the modeling. P20: *"He helped me by showing some details that had gone unnoticed"*, P4: *"He helped a lot by showing some things that I hadn't understood or answering questions"* and P23: *"His modeling covered some big flaws that were in mine."*
- **New ideas generation and Comparison**, it was stated that the LLM contributed to broaden the perspectives of students and enable comparison between different solutions. P26: *"Chat added new ideas, things I hadn't thought of"* P3: *"It brought a different perspective,"* and P17: *"It was a good experience for comparing the possibilities and differences between my diagram and the one in Chat. It also helps open the mind to a possible solution that the developer hadn't considered when creating their diagram."*

However, the students also pointed out some limitations and had some suggestions to improve the process.

- **Limitations in the definition of associations**, some students highlighted challenges that the LLM presented in establishing relations between the classes, as stated in: P7: *"he didn't convey much clarity when making associations between classes"*, P10: *"He didn't present the connections correctly in some cases."* and P25: *"he makes mistakes in some relations between classes."*
- **General Limitations**, it was also mentioned challenges related to the completeness and precision of the diagrams generated by ChatGPT: P13: *"ChatGPT didn't generate some important classes for the system nor did it write the names of the relations, which hindered the understanding of the diagram it generated."* P14: *"However, it's not reliable. There were some errors and disagreements in decisions when modulating."* P16: *"The diagram it generated was missing several things and disagreed with the statement and my diagram as well."*
- **Suggestion for the improvement of usage**, students suggested ways of implementing the tool to be more effective, such as: detailing and specifying the prompt used and use the diagram generated as a way to get more ideas and details, as stated in: P21: *"I believe it's an excellent tool to help create diagrams, but it's important to create a very detailed prompt and pay attention to small errors that may occur."*, P29: *"It helped. I wouldn't use it to create the entire diagram, but I would use it to get ideas of what to do. I'm often unsure if I'm on the right track. When ChatGPT creates the diagram, it ends up helping me see details or other ways of doing things that*

I hadn't thought of." and P1: "I believe that if we try a more specific prompt, it would generate a better diagram."

When asked if they would use the LLM-based process again for a similar modeling task, those that answered "In some cases" were questioned about these cases where they would consider the reuse of the tool. Some of the answers were: P1: ´´I would use it in complex projects to validate my ideas in specific parts of the diagram." and ´´A use case that I believe is good would be building the diagram iteratively.", P5: ´´Cases where I want to see another way of creating a diagram besides the one I did myself.", P7: ´´I would use it to create the classes and attributes, in addition to trying to use the gpt chat to generate the basic sketch of the diagram.' and P20: ´´Cases where I may have missed some crucial detail of the project. Obviously, this should be done at the beginning."

The qualitative data reveals a nuanced and dual perception of using an LLM for UML class modeling. Students overwhelmingly recognized the tool's value as an ideation partner and a scaffolding mechanism, adept at accelerating the initial stages of modeling, identifying forgotten components, and proposing alternative design solutions. However, this enthusiasm was consistently tempered by the tool's perceived unreliability, particularly concerning the correctness of relationships and the completeness of the models. The feedback suggests that the LLM is not viewed as an autonomous agent capable of producing a final diagram, but rather as a powerful collaborator.

5 Lessons Learned

Based on our experience conducting this study, we offer the following lessons for educators and researchers interested in leveraging LLMs for software modeling education.

Improve Prompt Design: Our study utilized a simple, direct prompt to establish a baseline for the LLM's generative capabilities. However, a recurring theme in the qualitative feedback was the student's perception that a more detailed prompt could yield better results. As P21 suggested, ´´it's important to create a very detailed prompt and pay attention to small errors". This indicates that the prompt itself is a key variable in the learning experience. Future research should explore the impact of structured versus open-ended prompts, or prompts that specify pedagogical roles (e.g., ´´Act as a Software Engineer tutor"). For educators, this means prompt design can be used as a mechanism to scale the difficulty of the task or to focus the learning on specific modeling concepts.

Using LLMs as a support: Our findings strongly suggest that the LLM is most effective when used as a tool to augment, not replace, a student's knowledge. The ability to critically analyze the LLM's output was directly proportional to the student's own understanding of UML principles. This refutes the idea that students will blindly accept AI-generated content. The key lesson for educators is that these tools should be introduced after students have acquired foundational knowledge. The LLM then becomes a powerful partner for practicing and refining skills, rather than a primary source of instruction.

Explore Iterative Use of LLMs: The methodology of our study was based on a ´´one-shot" interaction: the student created a diagram, the LLM created another, and the student compared them. While effective for fostering critical analysis, student feedback suggests a desire for a more dynamic interaction. As P1 stated, ´´A use case that I believe is good would be building the diagram iteratively." This points to a valuable direction for future studies: exploring a co-creation model where the student and the LLM engage in a dialogue, refining a single diagram over multiple turns. Such a process would more closely mirror a real-world pair programming or design review session and could yield different, potentially deeper, learning outcomes.

6 Limitations

This section describes limitations of the study which may affect the generalization and interpretation of the findings. We categorize these into threats to external and internal validity.

Threats to External Validity:

- **Sample Profile and Size:** Our study was conducted with a cohort of 31 undergraduate students from a single computer science program at one university. This specific demographic profile and limited sample size mean that the findings may not be directly applicable to students at other institutions, in different cultural contexts, or at different stages of their academic career.
- **Language of Intervention:** The entire study, including the prompts and scenarios, was conducted in Portuguese. While this was a deliberate choice to ensure comprehension and remove language proficiency as a confounding variable, it also means the results may not generalize to other languages. LLMs are known to have performance variations across different languages, and the effectiveness of the prompts could differ in an English-speaking context, for example.

Threats to Internal Validity:

- **Duration and Time Constraints:** The intervention was conducted over two 120-minute class periods, with the final assignment limited to 90 minutes. As some students noted in their feedback, this time constraint may have caused them to feel rushed, particularly during the final written analysis. This could have impacted the depth and quality of their qualitative reflections.
- **Construct Validity of Self-Assessment:** Our quantitative measure relied on students' self-assessment of their knowledge. While we observed a statistically significant increase, self-perception is not a direct measure of actual knowledge gain. The improvement could be partially attributed to an increase in confidence rather than a change in capability alone. Our analysis of the student-generated diagrams aimed to mitigate this, but it remains a limitation of the quantitative data.

7 Conclusion

This study was designed to investigate a novel LLM-based pedagogical approach for teaching UML class diagrams, guided by three primary objectives. Regarding the first objective, our quantitative analysis confirmed that the intervention had a statistically significant positive impact on students' self-perceived knowledge. For the second and third objectives, the analysis revealed that students perceive the LLM not as an oracle, but as a valuable "co-pilot´´,

useful for augmenting their design process through ideation and comparison.

These results have direct implications for software engineering education. They suggest that the most effective use of LLMs is not as a replacement for foundational instruction, but as an augmentative tool that fosters critical thinking. By framing the LLM as a ´´co-pilot," educators can encourage students to engage in a reflective comparison, strengthening their reasoning skills. This perspective shifts the learning objective from simply finding the correct diagram to building a well-justified diagram through collaboration with an AI partner.

While our study demonstrated the effectiveness of a one-shot comparative model, future research should explore more dynamic forms of interaction. Investigating iterative, co-creative processes where students and LLMs refine a single diagram together presents a promising avenue. Furthermore, research into pedagogical prompt engineering—designing prompts that can, for instance, simulate common student errors or guide Socratic dialogues—is a critical next step. Acknowledging the specific context of our study, we conclude that the integration of LLMs holds immense potential to transform design education, making it more interactive, personalized, and reflective.

ARTIFACT AVAILABILITY

All artifacts generated and used in this empirical study are publicly available in a permanent repository. The full dataset can be accessed via the following DOI: https://doi.org/10.6084/m9.figshare.29710610.

The repository includes the following materials:

- The full pre- and post-study questionnaires used for data collection;
- The complete set of anonymized student responses to both questionnaires;
- The slide decks for the "Introduction to PlantUML" and "LLMs for Modeling" lectures;
- The full assignment scenario, including the problem description;
- The complete set of anonymized student submissions from the assignment, including their generated diagrams and handwritten analyses (transcribed).

ACKNOWLEDGMENTS

We thank USES Research Group members for their support. We would like to thank the financial support granted by Project No 017/2024 Divulga CT&I FAPEAM; This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES-PROEX) - Finance Code 001. This work was partially supported by Amazonas State Research Support Foundation - FAPEAM - through the POSGRAD project 2025/2026. CNPq 314797/2023-8; CNPq 443934/2023-1; CNPq 445029/2024-2; Amazonas State University (UEA) through Academic Productivity Program 01.02.011304.026472/2023-87.

REFERENCES

[1] Sohail Alhazmi, Charles Thevathayan, and Margaret Hamilton. 2021. Learning UML Sequence Diagrams with a New Constructivist Pedagogical Tool: SD4ED. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 893–899. doi:10.1145/3408877.3432521

[2] Nilufar Baghaei and Antonija Mitrovic. 2006. A Constraint-Based Collaborative Environment for Learning UML Class Diagrams. In *Intelligent Tutoring Systems*, Mitsuru Ikeda, Kevin D. Ashley, and Tak-Wai Chan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 176–186.

[3] Lenz Belzner, Thomas Gabor, and Martin Wirsing. 2023. Large Language Model Assisted Software Engineering: Prospects, Challenges, andnbsp;anbsp;Case Study. In *Bridging the Gap Between AI and Reality: First International Conference, AISoLA 2023, Crete, Greece, October 23–28, 2023, Proceedings* (Crete, Greece). Springer-Verlag, Berlin, Heidelberg, 355–374. doi:10.1007/978-3-031-46002-9_23

[4] Alberto. Cardoso, Gustavo R.. Alves, and Teresa. Restivo. 2020. *Proceedings of the 2020 IEEE Global Engineering Education Conference (EDUCON) : date and venue, 27-30 April, 2020, Porto, Portugal.* IEEE.

[5] Jianguo Chen, Huijuan Lu, Lixin An, and Yongxia Zhou. 2009. Exploring teaching methods in software engineering education. In *2009 4th International Conference on Computer Science & Education*. IEEE, 1733–1738.

[6] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. 2023. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling* 22 (6 2023), 781–793. Issue 3. doi:10.1007/s10270-023-01105-5

[7] Alessio Ferrari, Sallam Abualhaija, and Chetan Arora. 2024. Model generation with LLMs: From requirements to UML sequence diagrams. In *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*. IEEE, 291–300.

[8] Eleonora Grassucci, Gualtiero Grassucci, Aurelio Uncini, and Danilo Comminiello. 2025. Beyond answers: How llms can pursue strategic thinking in education. *arXiv preprint arXiv:2504.04815* (2025).

[9] Bilal Karasneh, Rodi Jolak, and Michel RV Chaudron. 2015. Using examples for teaching software design: An experiment using a repository of uml class diagrams. In *2015 Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 261–268.

[10] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.

[11] Maryam Khan, Muhammad Azeem Akbar, and Jussi Kasurinen. 2025. Integrating LLMs in Software Engineering Education: Motivators, Demotivators, and a Roadmap Towards a Framework for Finnish Higher Education Institutes. *arXiv preprint arXiv:2503.22238* (2025).

[12] Ranim Khojah, Mazen Mohamad, Philipp Leitner, and Francisco Gomes de Oliveira Neto. 2024. Beyond code generation: An observational study of chatgpt usage in software engineering practice. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1819–1840.

[13] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50 – 60. doi:10.1214/aoms/1177730491

[14] Modeling-Languages.com. 2024. Textual UML Tools – The Complete List. https://modeling-languages.com/text-uml-tools-complete-list/.

[15] PlantUML Team. 2025. PlantUML: Create diagrams from textual description. https://plantuml.com/.

[16] Nishat Raihan, Mohammed Latif Siddiq, Joanna CS Santos, and Marcos Zampieri. 2025. Large language models in computer science education: A systematic literature review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*. 938–944.

[17] S. S. SHAPIRO and M. B. WILK. 1965. An analysis of variance test for normality (complete samples)†. *Biometrika* 52, 3-4 (12 1965), 591–611. doi:10.1093/biomet/52.3-4.591 arXiv:https://academic.oup.com/biomet/article-pdf/52/3-4/591/962907/52-3-4-591.pdf

[18] Ven Yu Sien. 2011. An investigation of difficulties experienced by students developing unified modelling language (UML) class and sequence diagrams. *Computer Science Education* 21 (12 2011), 317–342. Issue 4. doi:10.1080/08993408.2011.630127

[19] Williamson Silva, Igor Steinmacher, and Tayana Conte. 2019. Students' and instructors' perceptions of five different active learning strategies used to teach software modeling. *IEEE Access* 7 (2019), 184063–184077.

[20] Ann T S Taylor. 2011. Top 10 reasons students dislike working in small groups … and why i do it anyway. *Biochemistry and Molecular Biology Education* 39 (2011).