

# Scaling Assessment of Student Models with LLMs: Integrating Feedback into Practice

Maximilian Sölch

maximilian.soelch@tum.de  
Technical University of Munich  
Munich, Germany

Stephan Krusche

krusche@tum.de  
Technical University of Munich  
Munich, Germany

## Abstract

Automated assessment of student modeling tasks is difficult to scale because UML diagrams are open-ended, graphical, and highly contextual. This paper presents a production-ready extension of Athena, an open-source feedback system that integrates directly with the learning platform Artemis to provide human-in-the-loop support for modeling exercises.

The approach introduces ApollonUML, a domain-specific textual representation that improves LLM interpretability while preserving precise links to diagram elements. Combined with rubric-style grading instructions, the system generates contextualized, pedagogically aligned feedback suggestions that human graders can review, adapt, or reject within their existing grading workflow.

A large-scale retrospective observational validation study on authentic student submissions shows that LLM-based feedback can approximate human grading, with characteristic deviations such as generosity toward weaker work, harshness toward stronger work, and failure cases tied to prompt context and model complexity. The analysis further reveals inconsistencies in human grading that LLMs may help surface. These findings indicate that LLM-based assessment is capable of reducing human grader workload while maintaining oversight in modeling education.

## CCS Concepts

• **Social and professional topics** → **Student assessment**; • **Applied computing** → **Education**; • **Software and its engineering** → *Unified Modeling Language (UML)*.

## Keywords

Software Engineering, Modeling Education, UML, Large Language Models, Grading, Formative Feedback

### ACM Reference Format:

Maximilian Sölch and Stephan Krusche. 2026. Scaling Assessment of Student Models with LLMs: Integrating Feedback into Practice. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEET '26)*, April 12–18, 2026, Rio de Janeiro, Brazil. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3786580.3786985>

## 1 Introduction

Modeling is a fundamental activity in software engineering that supports abstraction, communication, and design analysis across the development lifecycle [10, 16, 28]. Unified Modeling Language (UML) diagrams are widely used in both industry and academia to express structural and behavioral aspects of software systems [32]. Educational research highlights the pedagogical value of teaching modeling early in computer science curricula to foster higher-level design thinking [1, 20].

Despite its importance, assessing modeling tasks at scale remains a challenge. Manual grading is time-consuming and introduces inconsistencies, as human graders may interpret solutions differently or apply criteria unevenly [22]. In large courses with hundreds of participants, this creates bottlenecks that delay feedback and reduce opportunities for iterative improvement. For students, delayed or inconsistent feedback can hinder progress and motivation [15]. For instructors, reviewing each submission individually makes it difficult to sustain high-quality feedback across multiple exercises.

While automated assessment techniques exist for code, the open-ended and graphical nature of UML models complicates assessment and feedback delivery. Existing approaches often focus on syntactic checks or similarity measures, which fall short of providing meaningful, pedagogically aligned feedback. This leaves educators with a difficult trade-off: either invest substantial resources into manual assessment or accept reduced feedback quality at scale.

To address this gap, this paper extends Athena [24], an open-source feedback system, with a module for automated assessment of UML submissions using large language models (LLMs). The goal is to support human graders by generating high-quality, contextualized feedback aligned with learning objectives. The proposed approach transforms student models into a domain-specific textual representation, incorporates exercise-specific grading criteria, and leverages LLMs to generate element-level feedback integrated into the modeling environment.

This paper presents a scalable, production-ready system for LLM-based feedback generation in modeling exercises. It introduces a pipeline that converts graphical UML models into structured prompts, enabling more effective interaction with LLMs. It proposes a strategy for incorporating structured grading criteria to align LLM output with pedagogical goals. An empirical evaluation compares LLM-generated to human grader feedback across multiple exercises with a focus on the following research questions (RQ):

**RQ1 Score Alignment:** To what extent can LLM-based assessment replicate human grading of student modeling submissions?

**RQ2 Failure Cases:** What are the limitations and failure cases of LLM-based modeling feedback generation?



This work is licensed under a Creative Commons Attribution 4.0 International License. *ICSE-SEET '26, Rio de Janeiro, Brazil*

© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2423-7/2026/04  
<https://doi.org/10.1145/3786580.3786985>

The remainder of the paper is structured as follows: Section 2 reviews related work on automated feedback and modeling assessment. Section 3 describes the functionality of the system, model representation, prompt design, and grading alignment. Section 4 presents the study setup and results, discusses key findings, limitations, and implications. Section 5 concludes and outlines directions for future work.

## 2 Related Work

Automated assessment of modeling tasks has been approached through diverse methods, ranging from rule-based and machine learning techniques to emerging LLM-based solutions. To contextualize this work, we review prior UML-specific approaches, recent LLM applications in modeling, and broader studies on LLMs for automated scoring.

### 2.1 Automated Assessment of UML Before LLMs

Early systems relied on rule-based comparison of student diagrams to instructor-provided solutions. UMLGrader [14] identified missing or erroneous elements in class diagrams by directly comparing them against a sample solution, offering formative feedback but only in tightly constrained exercises. UMLGrade [11] extended this idea by generating structured assessment reports based on semantic, syntactic, and design rule checks, aiming to reduce grading effort while reinforcing object-oriented design principles.

Other approaches focused on improving flexibility. A metamodel-based method [7] mapped student class diagrams to instructor reference models using syntactic, semantic, and structural criteria, enabling recognition of multiple valid solutions. Similarly, Fauzan et al. [12] combined semantic similarity (lexical information) and structural similarity (diagram topology), finding that experts weigh both dimensions equally and that the approach achieves reliability comparable to human assessors. Beyond class diagrams, Striwe and Goedicke [26] applied dynamic checks to UML activity diagrams using trace generation and sequence alignment, demonstrating how behavioral aspects can be evaluated automatically.

Several tools emphasized learner support. Thomas et al. [27] proposed a five-stage framework to grade student-drawn ER diagrams, addressing missing or malformed elements and achieving high agreement with human graders. Hoggarth and Lockyer [17] developed one of the earliest systems for interactive comparison of student diagrams with model solutions, prioritizing feedback for learning over grading automation. Similarly, COCLAC [6] transformed UML diagrams into Java code for testing, providing immediate correctness checks and linking modeling with programming practice.

To improve scalability, machine learning techniques were also explored. Krusche [18] presented a semi-automatic assessment system that learned from manually graded submissions to propose scores and feedback for new ones. Applied in a large course with over 800 students, it achieved up to 80% automation with low adjustment rates by human graders, improving both efficiency and consistency. In contrast, Stikkolorum et al. [25] trained regression and classification models to predict UML class diagram grades but reported limited predictive accuracy, highlighting the need for larger datasets and more robust features. Collectively, these pre-LLM approaches

show the promise of automation but reveal strong dependencies on predefined reference models, constrained solution spaces, or extensive training data.

### 2.2 LLM-based Assessment and Feedback for Modeling

With the advent of LLMs, new approaches have emerged that promise greater flexibility and contextualization. Ardimento et al. [2] introduced a retrieval-augmented generation (RAG) system integrated into a cloud-based modeling tool, which provided targeted feedback on more than 5,000 labeled UML diagrams. Extending this, UML Miner [3] combined process mining with RAG-based LLMs in a Visual Paradigm plugin, offering continuous, personalized feedback throughout the modeling workflow.

Other studies explored textualization strategies to make UML diagrams more interpretable for LLMs. Bouali et al. [8] converted visual class diagrams into textual descriptions and compared grades assigned by GPT o1-mini, Claude Sonnet, and human assessors. Their results showed strong alignment, with correlation coefficients above 0.76 and mean absolute errors below four points on a 40-point scale. Similarly, Wang et al. [30] applied GPT to evaluate use case, class, and sequence diagrams across 11 predefined criteria, reporting high but inconsistent alignment with human experts.

Critical perspectives also highlight limitations. Cámara et al. [9] documented semantic inconsistencies and scalability issues in ChatGPT's UML feedback, pointing to gaps in domain-specific reasoning. DUET [13], a prototype for UML and ER diagrams, employed a multi-stage LLM pipeline to generate reflective feedback. While praised for accessibility and scalability in interviews with educators, concerns remained around reliability and potential misuse. These works illustrate the potential of LLMs for modeling tasks but remain prototypes or small-scale studies, lacking production-level integration or systematic human-in-the-loop evaluation.

### 2.3 LLMs for Automated Scoring Beyond Modeling

Research outside UML provides additional evidence for the scoring capabilities of LLMs. In programming education, Mendonça et al. [21] compared LLaMA 3.2 and GPT-4o against human graders using rubric-based evaluations, finding that premium LLMs can produce grading patterns statistically indistinguishable from humans. In essay scoring, Xia et al. [33] tested ChatGPT on TOEFL writing tasks, showing operational feasibility but regression effects and prompt sensitivity. Broader comparisons between human and automated scoring highlight fairness concerns, particularly for English learners, where automated systems and human graders diverge [29].

At a larger scale, LLMs have also been studied as evaluators in NLP and software engineering tasks. JUDGE-BENCH [5] benchmarked 11 LLMs across 20 datasets, showing that reliability varied widely by dataset and evaluation property. Similarly, Wang et al. [31] found that prompting LLMs as evaluators could achieve near-human correlation on code translation and generation tasks, but performance was inconsistent across tasks. Together, these studies

show that LLMs can approximate human scoring in diverse contexts, while also underscoring the importance of domain adaptation and validation when applying them to modeling education.

Unlike prior work on rule-based comparison, ML-assisted grading, or recent LLM prototypes, this paper contributes a production-ready integration of LLM-based assessment into a widely used learning management system (LMS). The approach emphasizes human-in-the-loop workflows, large-scale evaluation on thousands of authentic submissions, and systematic analysis of grading alignment and deviations, highlighting both the opportunities and limitations of LLM-based modeling assessment.

### 3 Approach

To enable high-quality, automated feedback for UML modeling exercises, we extended the existing open-source<sup>1</sup> feedback system Athena with support for assessing modeling submissions. Athena is responsible for generating feedback and is integrated with the open-source<sup>2</sup> learning platform Artemis [19], which students use to submit their modeling solutions and review the feedback they receive.

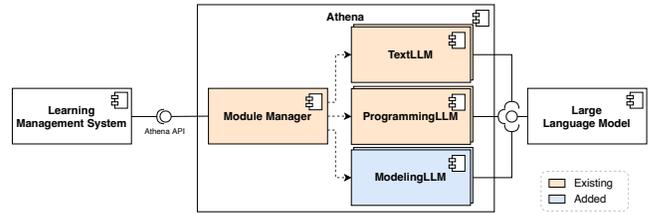
Artemis supports a variety of interactive exercise formats, including programming, quizzes, and modeling exercises. For modeling exercises, Apollon [20], an open-source<sup>3</sup> modeling editor, is embedded directly into the Artemis interface, allowing students to construct and submit their models seamlessly within the context of an exercise.

#### 3.1 Reference Implementation

The open-source system Athena serves as the reference implementation of the proposed approach. Athena is designed as an extensible feedback system that integrates with learning platforms such as Artemis to deliver automated feedback for multiple exercise types. For modeling exercises, Athena supports human graders during the manual assessment process by generating draft feedback suggestions for student-submitted models.

Figure 1 shows Athena’s top-level architecture and its integration points. In the evaluated deployment, Artemis, acting as the *Learning Management System*, receives student submissions and forwards them to Athena via a REST interface. Athena routes each request to the responsible feedback module and returns the generated feedback in the format required by Artemis for presentation in the assessment interface. Although the evaluation focuses on Artemis, Athena remains LMS-agnostic and exposes a standardized OpenAPI<sup>4</sup> interface that enables integration with other learning platforms.

Athena follows a modular microservice architecture. A central *Module Manager* handles all incoming feedback requests, verifies authorization, and forwards them to the appropriate modules. In addition to the modeling module introduced in this work, Athena provides dedicated modules for programming and text exercises (*ProgrammingLLM* and *TextLLM*).



**Figure 1: UML component diagram showing the top-level architecture of the reference implementation Athena.**

Following Athena’s microservice architecture, the modeling feedback component is realized as a separate module (*ModelingLLM*) that generates feedback for modeling submissions using LLMs. This design aligns with Athena’s modular structure, which allows independent deployment and scaling of feedback modules according to the needs of specific exercises or student cohorts.

Figure 2 describes the process of generating and delivering LLM-based feedback for modeling exercises. The process begins when a student creates and submits a UML model using Apollon embedded within Artemis. Upon submission, Artemis forwards the model to Athena for analysis. The feedback generation pipeline in Athena consists of three stages: formatting, predicting, and parsing.

In the formatting stage, Athena collects all relevant context information, including the exercise problem statement, an optional example solution, and any grading instructions defined by the instructor. Both the student’s model and the example solution are transformed into a domain-specific language (DSL) representation to improve LLM interpretability. These inputs are then injected into a predefined prompt template that guides the LLM’s response.

During the predicting stage, Athena invokes the configured LLM to generate a response based on the formatted prompt. The LLM returns textual feedback suggestions describing issues or areas for improvement in the submitted model.

In the parsing stage, Athena extracts individual feedback items from the LLM response and maps them to the feedback structure required by Artemis. This includes resolving references from the DSL representation back to the original model elements to enable inline feedback visualization. Feedback that can be linked to a specific model element is classified as referenced feedback, while general comments or suggestions about missing elements are considered unreferenced feedback.

After the feedback has been returned to Artemis, human graders are presented with the suggestions for review within the assessment interface. Feedback appears either inline—directly attached to the relevant model elements—or below the diagram, depending on whether it refers to specific parts of the model or more general aspects of the submission. Human graders can review, modify, or discard any feedback item before finalizing the assessment. Figure 3 illustrates how referenced feedback is anchored inline to a specific model element within Apollon, allowing for immediate contextual review.

General feedback, shown in Figure 4, appears beneath the modeling canvas and typically addresses missing elements or provides high-level guidance. From the student’s perspective, the interface mirrors this structure, but omits the assessment controls for modifying or approving feedback.

<sup>1</sup><https://github.com/ls1intum/Athena>

<sup>2</sup><https://github.com/ls1intum/Artemis>

<sup>3</sup><https://github.com/ls1intum/Apollon>

<sup>4</sup><https://www.openapis.org>

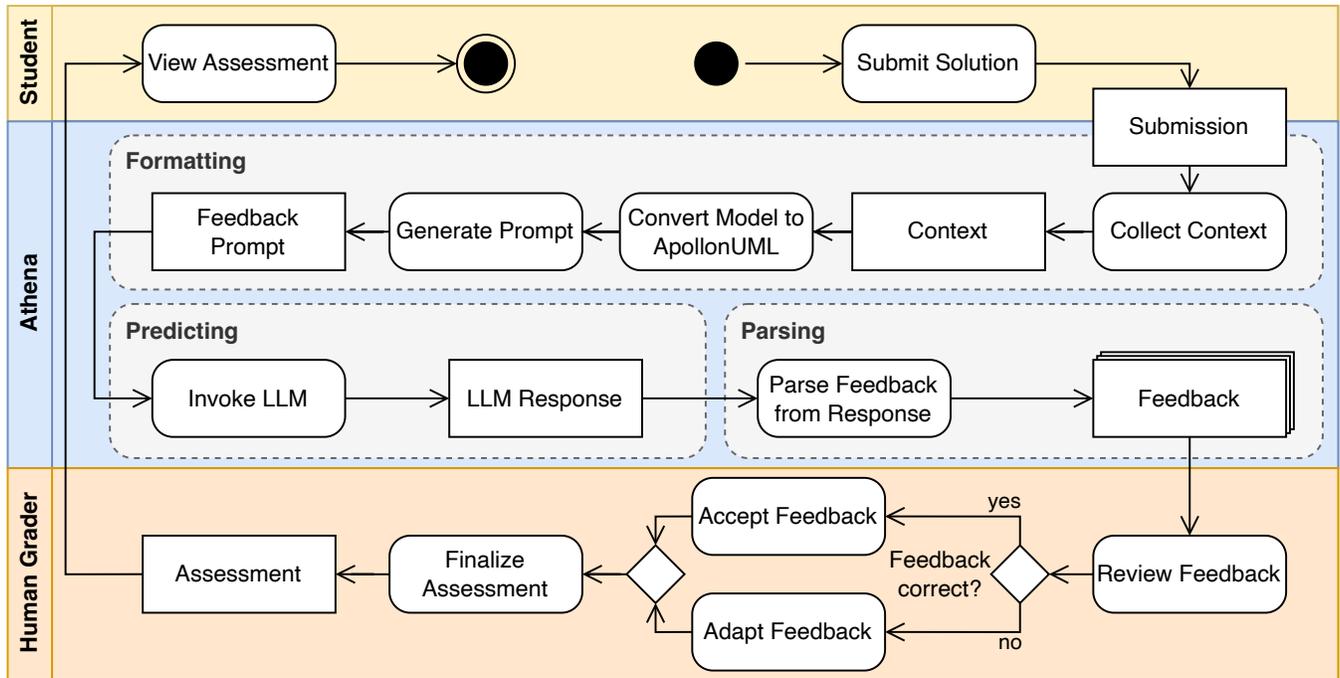


Figure 2: UML activity diagram visualizing the process of generating and delivering LLM-based feedback for modeling exercises with Athena.

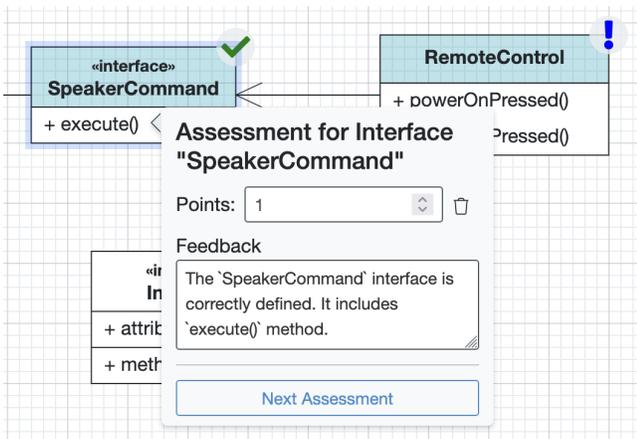


Figure 3: Referenced feedback suggestion anchored to a specific model element in the assessment interface of Artemis.

### 3.2 Model Representation

To prepare UML submissions for LLM-based feedback generation, Athena transforms student-created models into a concise, semantically explicit textual representation. Although Apollon internally stores diagrams in a structured JSON format, this format is not well suited for direct use with large language models, as it is verbose, nested, and tightly coupled to the internal structure of the editor. To improve interpretability and reduce prompt complexity, we convert each model into a DSL called ApollonUML, which serves as

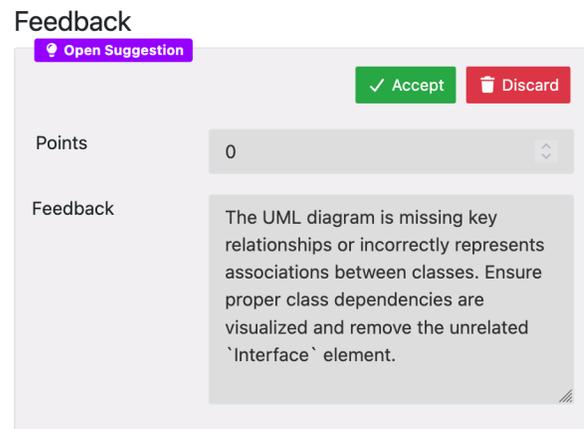


Figure 4: General, unreferenced feedback suggestion in the assessment interface of Artemis.

an intermediate representation between the diagram and the LLM prompt.

ApollonUML preserves the essential structure and semantics of the original UML diagram while expressing elements and relationships in a compact textual format. The syntax draws inspiration from notations such as PlantUML<sup>5</sup> and Mermaid<sup>6</sup>, but it avoids symbolic shorthand in favor of fully spelled-out constructs

<sup>5</sup><https://plantuml.com>

<sup>6</sup><https://mermaid.js.org>

to improve clarity. This explicit format reduces the likelihood of misinterpretation by the LLM and simplifies prompt engineering. Internally, the transformation pipeline parses the Apollon JSON and extracts diagram elements, relationships, and their hierarchies. It also maintains a mapping between elements in the DSL and their corresponding diagram identifiers to support feedback anchoring and referencing.

To ensure naming consistency, the DSL includes a name disambiguation mechanism. When multiple elements share the same name, suffixes (e.g., #A, #B) are appended to distinguish them. Elements without a name receive placeholder identifiers (e.g., ##A). This guarantees uniqueness in the representation while preserving human readability in the LLM prompt.

The DSL supports all diagram types currently available in Apollon, including various forms of UML, as well as BPMN models and Petri nets. An example of the ApollonUML format is shown in Listing 1.

```
UML Diagram Type: [type]

@Elements:
[type] ElementName {
  attributes:
    attributeName: type
  methods:
    methodName(parameters): returnType
  marker: [markerType]
}

@Relations:
R1: SourceElement (relationType) --> TargetElement: label {
  SourceElement: {
    role: roleLabel
    multiplicity: multiplicityValue
  }
  messages: [
    { name: messageName, to_direction: targetElement }
  ]
}

@Owners:
OwnerElement: ChildElement1, ChildElement2
```

**Listing 1: Structure and syntax of the ApollonUML format used to represent UML diagrams textually for LLM input.**

### 3.3 Grading Instructions

To generate consistent and pedagogically aligned feedback, Athena incorporates structured grading instructions defined within the Artemis platform. These grading instructions act as rubric-based criteria that guide both human graders and the LLM in evaluating student submissions. Each instruction includes predefined feedback, associated points, and optionally examples, allowing instructors to specify exactly what constitutes correct, incorrect, or partially correct modeling constructs. Although students do not see these instructions directly, they receive feedback that reflects the applied criteria whenever a human grader or Athena attaches a corresponding grading instruction to a specific model element.

When generating LLM-based feedback, Athena includes the relevant grading instructions in the system prompt. This ensures that the LLM’s output aligns with the instructor’s expectations and the learning goals of the specific exercise. Without these structured

instructions, the LLM would default to generating generic feedback, which may be educational but does not necessarily reflect the intended evaluation criteria.

To further improve consistency and reduce instructor workload, Athena automatically generates structured grading instructions if the instructor did not define any when setting up the exercise in Artemis. For this functionality, Athena uses a dedicated LLM prompt, which is shown in Listing 2, to transform the problem statement, example solution, and any informal grading notes into a machine-readable rubric. The resulting instructions serve as a high-quality starting point for manual refinement or direct use and follow a detailed, criteria-based format to ensure alignment with the pedagogical goals of the exercise.

You are an AI tutor for **{submission\_uml\_type}** modeling exercise assessment at a prestigious university.

Create a structured grading instruction based on the given grading instructions (important), the solution diagram (if present) and the problem statement. The structured grading instruction should be highly detailed to ensure consistent grading across different tutors.

<-Exercise Problem Statement->

**{problem\_statement}**

<-Grading Instructions->

Max points: **{max\_points}**, bonus points: **{bonus\_points}**

Instructions:

**{grading\_instructions}**

<-Official Example Solution->

**{example\_solution}**

Please return the structured grading instructions in the correct output json format.

**Listing 2: Prompt for generating structured grading instructions.**

### 3.4 Feedback Prompt Design

To generate high-quality, contextualized feedback for modeling submissions, Athena constructs prompts that incorporate a rich set of information specific to the exercise and the student’s solution. Each prompt includes the exercise problem statement, an example solution (if available), and the structured grading instructions defined by the instructor. This information enables the LLM to generate feedback that is not only technically accurate, but also aligned with the pedagogical intent of the assignment.

Athena uses a single-shot prompt structure based on the Chat Completions API format, consisting of a system message that defines the task and a human message that provides the input. The system message frames the LLM as an AI tutor responsible for providing feedback a human grader would accept, and explicitly lists characteristics that the feedback should fulfill, such as being constructive, specific, and educational. The human message includes the student’s submission in a domain-specific textual format and instructs the LLM to return feedback in a predefined JSON format.

Listing 3 shows the template used as the system message when generating feedback for modeling exercises. Notably, the prompt also includes structured grading instructions in a machine-readable JSON format, allowing the LLM to reason over grading criteria explicitly. This integration helps align the generated feedback with

the expectations set by the instructor, improves consistency across assessments, and supports point-based evaluation logic.

Athena is designed to be LLM-agnostic and supports a range of language models through a unified API layer based on LangChain<sup>7</sup>. This enables flexible deployment across cloud-based and self-hosted models, including OpenAI models<sup>8</sup>, Anthropic’s Claude<sup>9</sup>, or open-source models such as Llama<sup>10</sup>, depending on institutional constraints and privacy requirements.

```

You are an AI tutor for {submission_uml_type} modeling exercise assessment at a
prestigious university.

Create graded feedback suggestions for a student’s {submission_uml_type}
modeling submission that a human tutor would accept.
Meaning, the feedback you provide should be applicable to the submission with
little to no modification.

<Feedback Style>
1. Constructive
2. Specific
3. Balanced
4. Clear and Concise
5. Actionable
6. Educational
7. Contextual

<Exercise Problem Statement>
{problem_statement}

<Grading Instructions>
Max points: {max_points}, bonus points: {bonus_points}

Instructions:
{structured_grading_instructions}

<Official Example Solution>
{example_solution}

Important:
- Make sure to provide detailed feedback for each criterion. Always try to be
specific as possible.
- Also make sure your feedback adds up to the correct number of points. If there
are n points available and everything is correct, then the feedback should add up to
n points.
- Deeply think about the diagram and what the student potentially missed,
misunderstood, or mixed up.
- For the `element_name` field in the output, reference the specific diagram
element, attribute, method, or relation related to the feedback. Use the following
formats:
  - For classes or elements: `<ClassName>`
  - For attributes: `<ClassName>.<AttributeName>`
  - For methods: `<ClassName>.<MethodName>`
  - For relations: `R<number>` (e.g., `R1`, `R2`)
- If the feedback is not related to a specific element, leave the `element_name`
field empty.

<UML Diagram Format>
The submission uses the following UML diagram format:
{uml_diagram_format}
- Note: Don't mention elements that have no name, by their artificial name: e.g. ##A
or ##B, instead just say e.g. the task in ... is missing ...

```

**Listing 3: Prompt template for generating feedback for modeling exercises.**

<sup>7</sup><https://www.langchain.com/>

<sup>8</sup><https://openai.com/>

<sup>9</sup><https://www.anthropic.com/>

<sup>10</sup><https://llama.meta.com>

## 4 Evaluation

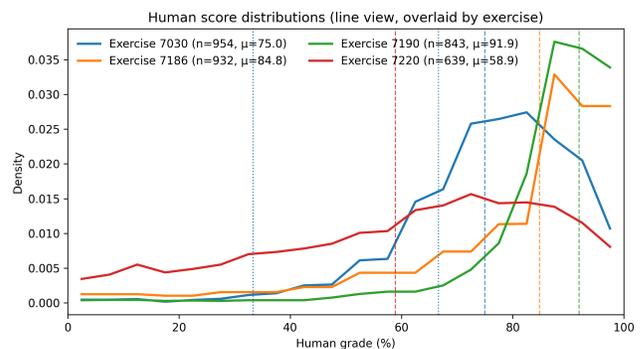
This section outlines the methodology used to evaluate the effectiveness of LLM-based feedback generation in the context of UML modeling exercises, as implemented in Athena. The evaluation focuses on assessing how closely LLM-generated scores align with human grading across real student submissions. It presents the study design, quantitative and qualitative results, a summary of key findings, a discussion of the implications, and a reflection on potential limitations.

### 4.1 Study Design

Following recent guidelines for empirical studies with large language models [4], the evaluation consists of a *retrospective observational validation study*, similar to approaches used in previous work [8, 21]. This design was appropriate because all data (student submissions and human-assigned scores) already existed prior to the study, and no experimental manipulation was applied. The study empirically assesses the degree of alignment between LLM-generated scores and human grading in authentic educational practice.

We address **RQ1 (Score Alignment)** through quantitative error analysis comparing human and LLM scores and **RQ2 (Failure Cases)** through a qualitative categorization of extreme score deviations.

To empirically evaluate the scoring accuracy of LLM-generated feedback, the evaluation includes a retrospective observational validation study using student submissions from four modeling exercises in an undergraduate software engineering course with up to 2,000 students. These exercises cover both UML class and activity diagrams and vary in complexity and point value. An overview is provided in Table 1, including the number of submissions per exercise, average model size, and average human-assigned scores. The distribution of human-assigned scores per exercise is shown in Figure 5.



**Figure 5: Distribution of human-assigned scores per exercise.**

All four exercises had instructor-defined grading instructions in Artemis. Therefore, Athena did not generate any grading instructions automatically. Instead, the LLM-based feedback generation relied directly on the provided rubric-style criteria, ensuring consistency with the human graders’ assessment setup.

**Table 1: Overview of the four modeling exercises used for the evaluation.**

Id	Title	Diagram	Points	Submissions	Elements (avg.)	Relations (avg.)	Score (avg. %)
7030	H07E02 Model the Strategy Pattern	UML Class Diagram	5	954	15.0	6.3	75.0
7186	L09E02 Model an Activity Diagram	UML Activity Diagram	4	932	13.3	13.9	84.8
7190	H09E02 Model a Food Delivery Process	UML Activity Diagram	8	843	25.8	30.5	92.0
7220	H10E02 Continuous Deployment	UML Class Diagram	10	639	23.7	8.5	58.9

The evaluation included all submissions without sampling or filtering. The original human grader assessments in Artemis serve as the baseline for comparison. In these four exercises, trained and experienced students who had previously completed the course served as human graders and carried out the assessments. LLM-based feedback was generated in August 2025 using Athena configured with *GPT-4o*, deployed via a private Azure OpenAI instance.

The system processed each submission, generated feedback, and derived a score based on the instructor-defined grading instructions. To capture performance and operational metrics, all LLM interactions were instrumented and traced using Langfuse<sup>11</sup>, which recorded end-to-end latencies, token usage, and costs for every submission.

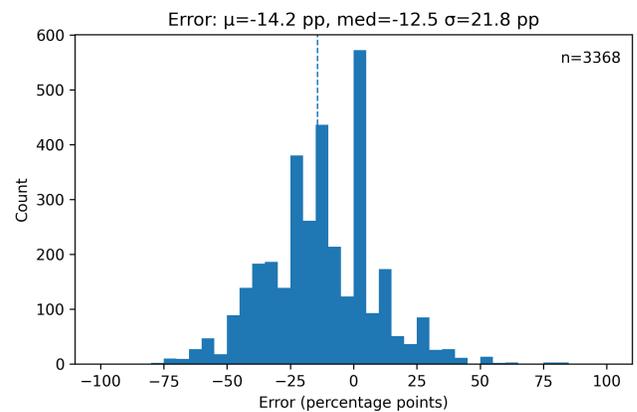
The evaluation is quantitative and measures the difference between LLM-generated scores and human-assigned scores. The analysis reports the error per submission (defined as LLM score – Human score) and analyzes the distribution of errors using descriptive statistics, including mean, median, and mean absolute error (MAE).

In addition to the quantitative error analysis, we conducted a qualitative inspection of the most contradicting submissions, defined as those with an absolute score deviation of at least 70 percentage points (pp) between human grader and LLM assessment. A total of 26 out of the 3,368 submissions fell into this category. To better understand the underlying causes, each case was categorized into one of three error categories: *LLM Overgrading*, *LLM Undergrading*, or *Mutual Misgrading*. *LLM Overgrading* refers to cases where the LLM assigned substantially higher scores than warranted, while *LLM Undergrading* captures cases where otherwise valid work was evaluated too harshly. *Mutual Misgrading* denotes situations in which both human grader and LLM scores deviated noticeably from the desired grading quality of the submission.

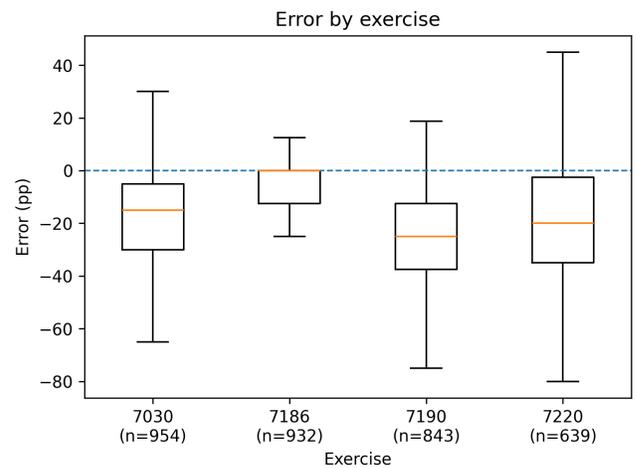
## 4.2 Results

Across all 3,368 submissions, the mean score assigned by human graders was 78.90%, while the LLM-generated scores averaged 64.68%. The overall mean absolute error (MAE) between human and LLM scores was 20.21 percentage points (pp), with a 95% bootstrap confidence interval of [19.67, 20.76], and the median absolute error was 18.75 pp (95% bootstrap CI [15.00, 18.75]). On average, the LLM assigned lower scores than human graders, as indicated by a mean error of -14.22 pp (95% CI [-14.95, -13.48]) and a median error of -12.50 pp. A paired t-test confirmed a systematic undergrading tendency of the LLM ( $t(3367) = -37.91, p < .001$ ), with a moderate standardized effect size ( $d_z = -0.65$ , 95% bootstrap CI [-0.69, -0.61]); this result was robust under a Wilcoxon signed-rank test ( $p < .001$ ).

Figure 6a provides a histogram of the overall errors (LLM – Human) across all four exercises. The distribution shows a skew toward negative values, indicating that the LLM more often underestimates errors compared to human grading. Figure 6b shows a box plot of errors (LLM – Human) per exercise.



(a) Distribution of errors (LLM – Human) across all 3,368 submissions. The vertical dashed line marks the mean error.



(b) Box plots of errors (LLM – Human) per exercise. The dashed line at 0 represents perfect agreement.

**Figure 6: Error analysis of LLM-generated scores compared to human grading.** (a) shows the overall distribution across all submissions, while (b) highlights variation between exercises.

<sup>11</sup><https://langfuse.com>

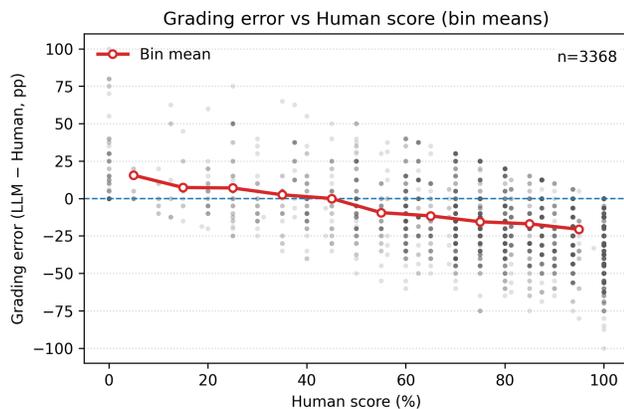
The per-exercise error metrics show notable variation: For Exercise 7030, the mean absolute error was 21.34 pp, with a median absolute error of 20.0 pp. The mean error was -13.86 pp, and the median error -15.0 pp. For Exercise 7186, the MAE was 12.59 pp, with a median absolute error of 12.5 pp. The mean error was smaller at -3.0 pp, and the median error was 0.0 pp, indicating a close alignment. Exercise 7190 had the highest observed deviation, with an MAE of 25.35 pp, a median absolute error of 25.0 pp, a mean error of -24.47 pp, and a median error of -25.0 pp. Finally, Exercise 7220 resulted in an MAE of 22.87 pp, a median absolute error of 20.0 pp, a mean error of -17.58 pp, and a median error of -20.0 pp.

Beyond grading accuracy, we also examined the efficiency and cost of generating feedback. Table 2 reports the end-to-end latency distribution and model usage costs per exercise. Median generation times ranged from 3.7 to 6.7 seconds per submission, with p95 latencies between 5.2 and 11.8 seconds. Overall costs were modest: the average expense per submission remained below two cents, with token consumption between 3,300 and 4,800 tokens on average. These results indicate that automated feedback generation is both time efficient and economically feasible at scale.

**Table 2: Cost and end-to-end (trace) latency per exercise. Latencies in seconds.**

Exercise	p50	p90	p95	p99	Total \$	\$/Subm.	Tok/Subm.
7030	5.060	7.399	8.234	10.049	13.95	0.01462	4073
7186	3.682	4.777	5.409	8.088	10.81	0.01160	3341
7190	3.842	4.801	5.217	6.407	13.53	0.01605	4872
7220	6.704	10.306	11.772	14.928	11.09	0.01736	4628

Figure 7 plots the grading error (defined as LLM score minus human score) against the human-assigned score, with binned mean residuals shown in red. Each point represents a single submission, while the red curve illustrates the average error within 10-percentage-point bins of human scores. For the first five bins (0-50%), the LLM tends to assign higher scores, while for higher human scores (above 50%), the LLM consistently assigns lower scores compared to human graders.



**Figure 7: Residuals of LLM-generated scores compared to human scores across all exercises.**

To better understand extreme deviations, we qualitatively analyzed the 26 submissions with an absolute score difference of at least 70 percentage points. Each case was reviewed from an instructor perspective by comparing the feedback provided by human graders and the LLM, and then categorized into one of three error types: *LLM Overgrading*, *LLM Undergrading*, or *Mutual Misgrading*.

The distribution in Table 3 shows that the most frequent failure case was *LLM Undergrading*, where valid submissions received scores that were substantially underestimated. Typical cases included otherwise correct models with only minor omissions (e.g., a missing method or a small syntax error) that the LLM penalized disproportionately. Another common pattern was *Mutual Misgrading*, in which both human and LLM scores deviated substantially from the desired grading quality. For instance, some diagrams received full credit from human graders despite evident flaws, while the LLM simultaneously graded them overly harshly. In several cases of *LLM Overgrading*, the LLM assigned high or even full scores to empty submissions by mistakenly referencing elements from the example solution provided in the prompt; only in one out of six such cases did the LLM correctly assign zero points.

The breakdown by exercise further illustrates variation in failure cases. Exercise 7190 showed only undergrading cases, often involving complex activity diagrams where the LLM penalized overlapping or unused elements too severely, despite the submissions being of high overall quality. By contrast, Exercises 7030 and 7186 displayed a more mixed distribution across all three categories: in 7030, human graders sometimes awarded excessive points for incomplete class diagrams while the LLM responded with very low scores, whereas in 7186, the LLM overgraded empty or flawed activity diagrams and undergraded others with only minor issues. Overgrading was particularly pronounced in Exercise 7220 due to multiple empty object diagrams being credited with substantial scores, while more complete submissions with small structural mistakes were often scored far too low.

**Table 3: Categorization of the 26 submissions with  $\geq 70$  pp score deviation, showing variation across exercises.**

Exercise	LLM Overgrading	LLM Undergrading	Mutual Misgrading
7030	2	2	3
7186	3	4	3
7190	0	4	0
7220	2	1	2
<b>Total</b>	<b>7</b>	<b>11</b>	<b>8</b>

### 4.3 Findings

The results of the quantitative evaluation show that LLM-generated scores exhibit moderate alignment with human grading across the four modeling exercises. The overall mean absolute error (MAE) was 20.21 percentage points (pp), with a median absolute error (MdAE) of 18.75 pp. The LLM consistently assigned lower scores than human graders on average, as reflected in a negative mean error of -14.22 pp.

Exercise 7186 showed the closest alignment with human scores (MAE = 12.59 pp), while Exercise 7190 had the largest discrepancies (MAE = 25.35 pp). Interestingly, these two exercises also differ in

structural complexity. Exercise 7186 had relatively low complexity, with an average of 13.3 elements and 13.9 relationships per submission. In contrast, Exercise 7190 was the most complex, averaging 25.8 elements and 30.5 relationships. This observation suggests that structural complexity may negatively impact the LLM's ability to generate accurate scoring.

A residual analysis revealed that the LLM tends to overestimate scores for low-performing submissions and underestimate them for higher-performing ones. While this underlines limitations in replicating exact human grader scores, the observed alignment is sufficient to consider LLM-generated assessments as useful drafts for human review.

**Main Finding for RQ1 (Score Alignment):** LLM-generated scores moderately align with human grading (MAE = 20.21 pp), supporting their use as draft assessments to accelerate human grader workflows.

Beyond this general alignment, the analysis also uncovered a systematic factor influencing LLM performance: the structural complexity of the diagrams. More complex submissions tended to yield larger deviations, suggesting that model size and intricacy can challenge the LLM's reasoning capacity.

**Additional Finding for RQ1 (Score Alignment):** Structural complexity correlates with LLM performance: more complex diagrams with many elements and relationships tend to produce larger score deviations.

Beyond numeric score differences, we inspected cases with extreme deviations to better understand potential failure cases. Specifically, we analyzed the 26 submissions with an absolute score deviation of at least 70 percentage points and categorized them into three recurring error types: *LLM Overgrading*, *LLM Undergrading*, and *Mutual Misgrading*. The most critical failure case was the misclassification of empty submissions, where the LLM often attributed example solution content (included in the prompt) to students' work. Only in one out of six empty-submission cases did the LLM correctly assign zero points. Other discrepancies were caused by overly harsh LLM grading of otherwise valid submissions, as well as situations where both human grader and LLM deviated from plausible scores. In contrast, some smaller deviations arose from overly generous or inconsistent human grading — for example, submissions with evident structural issues that still received full credit. These observations suggest that structural complexity interacts with failure cases: more complex exercises such as 7190 were especially prone to undergrading, whereas simpler ones still showed risks of overgrading when empty submissions occurred.

**Main Finding for RQ2 (Failure Cases):** Major score deviations were caused by prompt limitations (e.g., misgrading empty submissions) and systematic LLM misgrading, with additional inconsistencies in human grading.

## 4.4 Discussion

The evaluation demonstrates that LLM-generated feedback can serve as a valuable support mechanism in the assessment of UML modeling exercises. While the observed alignment between LLM and human scores is not sufficient for fully automated grading in high-stakes scenarios, it is strong enough to meaningfully reduce human grader workload. In practice, the system allows human graders to focus their attention on refining and validating feedback rather than producing it from scratch. This shift in responsibility may foster deeper instructor-student interactions and more consistent assessment practices.

At the same time, the evaluation highlights clear limitations in fully relying on LLMs for assessment. Certain failure cases, such as scoring empty submissions, reveal that the current prompt design does not sufficiently constrain the model's behavior in this case. These limitations stress the importance of maintaining human oversight, particularly in edge cases, and suggest the need for more robust prompt engineering and input validation mechanisms.

Another issue arises from the impact of structural complexity on LLM performance. More complex models with many elements and relationships tended to produce larger score deviations, indicating that the current configuration struggles with reasoning over intricate structures. This issue may be mitigated by improving how context is presented to the model, for example through more compact prompt representations, selective inclusion of salient elements, or enhanced framing strategies that help the LLM focus on the most relevant aspects of the diagram.

Interestingly, some of the largest score deviations were not due to LLM faults but rather inconsistencies in human grading. In several cases, human graders awarded full credit to submissions with visible modeling flaws, whereas the LLM appropriately deducted points. This observation raises the possibility of using LLM-generated feedback to support human grader calibration or identify outliers in grading behavior.

The design of the prompt and the way context is framed within it play a critical role in ensuring useful output. Including both the problem statement and the example solution provides the model with rich context, but also introduces risks—most notably the LLM conflating example content with student work. This issue was particularly evident in the contradicting cases, where several empty submissions received high scores because the model mistakenly assessed the example solution instead. Future versions of the prompt should therefore more clearly delineate submission content from example material to avoid such confusion. In addition, empty submissions could be detected prior to invoking the LLM, preventing this systematic error altogether while simultaneously reducing unnecessary computation costs.

Beyond accuracy, interpretability and trustworthiness of feedback are crucial for practical adoption. The integration with Artemis ensures that referenced feedback is presented inline within the student's model, allowing human graders to quickly verify its relevance and correctness. This visual anchoring of feedback supports efficient review and correction, potentially increasing the trust of human graders and reducing cognitive load.

Finally, Athena’s open-source foundation enables seamless integration with existing educational platforms while allowing institutions to adapt the system to their specific needs. Its modular design supports experimentation with different LLMs or feedback strategies without major implementation overhead, ensuring that the approach can evolve alongside new pedagogical requirements and technological advancements.

Overall, while limitations remain, the proposed system demonstrates strong potential for enhancing feedback workflows in modeling education. With further refinement, the accuracy of LLM-generated feedback can be improved, reducing error rates and enabling more reliable support for both summative and formative assessment across diverse modeling scenarios.

#### 4.5 Limitations

To outline the limitations of the conducted evaluation, we adopt the categorization framework proposed by Runeson and Höst [23], addressing potential threats to internal, external, and construct validity as well as reliability:

*Internal Validity:* The reference scores used in the evaluation were provided by individual human graders without peer review or double marking. This introduces the risk of subjective bias or inconsistency, potentially affecting the reliability of the ground truth against which the LLM was compared. Additionally, the evaluation used the default temperature and sampling parameters of GPT-4o, without controlling for stochastic variability in the model’s output. This may limit the reproducibility of specific results.

*External Validity:* The evaluation was conducted on data from a single course at one university, which may limit the generalizability of the results to other contexts, such as different institutions, education levels, or domains. While Athena supports a variety of UML diagram types, the evaluation was limited to class and activity diagrams. It remains unclear how the system performs on other diagram types, such as sequence or component diagrams. Furthermore, only GPT-4o was used as the backend model. The generalizability of the results to other LLMs—such as Claude, LLaMA, or DeepSeek—was not assessed. Finally, human graders did not interact with the LLM-generated feedback during actual grading. Thus, the real-world impact of the system on human grader workflows or student learning outcomes was not part of this evaluation.

*Construct Validity:* The evaluation focused exclusively on the alignment of overall scores between LLM and human grading. However, score similarity does not necessarily reflect the pedagogical value, specificity, or usefulness of the generated feedback. No qualitative analyses or user studies with human graders or students were conducted to assess these aspects. Similarly, the extent to which the feedback aligns with intended learning outcomes remains unexplored.

*Reliability:* The evaluation pipeline, including data loading, metric computation, and figure generation, is fully scripted in automated evaluation scripts and included in the replication package, together with all relevant input data (student submissions, tutor feedback, and LLM-generated feedback), enabling exact re-execution. However, LLM outputs are inherently non-deterministic, so regenerating feedback may yield slightly different results unless the original outputs are reused. Only the categorization of failure cases

relied on manual annotation and may therefore introduce minor subjectivity.

## 5 Conclusion

This paper presented a production-ready, human-in-the-loop system for automated feedback on UML modeling tasks that integrates tightly with Artemis via Athena. A central contribution is the domain-specific textual representation ApollonUML, which improves LLM interpretability, enables precise anchoring of feedback to diagram elements, and applies uniformly across all supported modeling types rather than being tied to a single notation. In a retrospective study on 3,368 authentic submissions across four exercises, LLM-generated scores showed moderate alignment with human grading (MAE = 20.21 pp; MdAE = 18.75 pp). The evaluation revealed systematic deviations—LLMs often graded weaker submissions too generously and stronger submissions too harshly—which were closely linked to failure cases such as empty or structurally complex models. Notably, some inconsistencies also stemmed from human grading, underlining the value of LLMs not only as an automation aid but also as a potential calibration tool for human graders. The system further proved efficient and economical (median latency 3.7–6.7s; average cost under \$0.02 per submission), supporting its use at course scale.

These results indicate that LLM-based assessment is ready to accelerate human grader workflows at scale, provided human oversight remains in place. The most pressing improvements are minor prompt and input adjustments to prevent critical errors, for example by better separating example material from student content and detecting empty submissions before invoking the model. Building on these refinements, we plan to evaluate human grader acceptance in a field study, examining how often feedback suggestions are adopted, adapted, or rejected. Possible directions for further work include testing robustness on high-complexity diagrams, conducting evaluations with LLMs beyond GPT-4o, investigating the quality of automatically generated rubrics, and studying the longer-term impact on student learning and feedback literacy. Overall, this work provides a concrete step toward reliable and scalable feedback in modeling education, bridging the gap between research prototypes and practical deployment in large courses.

## Acknowledgments

A replication package including the datasets, generated feedback, and code for the evaluation discussed in this paper is available on OSF<sup>12</sup>. The underlying systems Athena, Artemis, and Apollon are available as open-source projects on GitHub and are therefore not duplicated in the replication package. ChatGPT was utilized to generate sections of this work, including text, tables, and code, as well as to revise sections to enhance clarity and readability. We have carefully reviewed all AI-generated content.

<sup>12</sup>[https://osf.io/vmc89/overview?view\\_only=19a2879ee54542cd912ed3f1f626e053](https://osf.io/vmc89/overview?view_only=19a2879ee54542cd912ed3f1f626e053)

## References

- [1] Luciane T. W. Agner, Timothy C. Lethbridge, and Inali W. Soares. 2019. Student Experience with Software Modeling Tools. *Software & Systems Modeling* 18, 5 (2019), 3025–3047.
- [2] Pasquale Ardimento, Mario Luca Bernardi, and Marta Cimitile. 2024. Teaching UML Using a RAG-based LLM. In *2024 International Joint Conference on Neural Networks (IJCNN)*, 1–8.
- [3] Pasquale Ardimento, Mario Luca Bernardi, Marta Cimitile, and Michele Scalera. 2024. A RAG-based Feedback Tool to Augment UML Class Diagram Learning. In *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS Companion '24)*. Association for Computing Machinery, 26–30.
- [4] Sebastian Balthes, Florian Angermeier, Chetan Arora, Marvin Muñoz Barón, Chunyang Chen, Lukas Böhme, Fabio Calefato, Neil Ernst, Davide Falessi, Brian Fitzgerald, Davide Fucci, Marcos Kalinowski, Stefano Lambiase, Daniel Russo, Mircea Lungu, Lutz Prechelt, Paul Ralph, Rijnard van Tonder, Christoph Treude, and Stefan Wagner. 2025. Guidelines for Empirical Studies in Software Engineering Involving Large Language Models. arXiv:2508.15503 [cs]
- [5] Anna Bavaresco, Raffaella Bernardi, Leonardo Bertolazzi, Desmond Elliott, Raquel Fernández, Albert Gatt, Esam Ghaleb, Mario Giulianelli, Michael Hanna, Alexander Koller, Andre Martins, Philipp Mondorf, Vera Neplenbroek, Sandro Pezzelle, Barbara Plank, David Schlangen, Alessandro Suglia, Aditya K Surikuchi, Ece Takmaz, and Alberto Testoni. 2025. LLMs Instead of Human Judges? A Large Scale Empirical Study across 20 NLP Evaluation Tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 238–255.
- [6] Philip-Daniel Beck, Thomas Mahlmeister, Marianus Ifland, and Frank Puppe. 2015. COCLAC - Feedback Generation for Combined UML Class and Activity Diagram Modeling Tasks. In *2. Workshop "Automatische Bewertung von Programmieraufgaben" (ABP'2015)*.
- [7] Weiyi Bian, Omar Alam, and Jörg Kienzle. 2021. Automated Grading of Class Diagrams. In *Proceedings of the 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS '19 Companion)*. IEEE Press, 700–709.
- [8] Nacir Bouali, Marcus Gerhold, Tosif Rehman, and Faizan Ahmed. 2025. Toward Automated UML Diagram Assessment: Comparing LLM-Generated Scores with Teaching Assistants. In *17th International Conference on Computer Supported Education*. 158–169.
- [9] 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, 3 (2023), 781–793.
- [10] A. J. Cowling. 2005. The Role of Modelling in the Software Engineering Curriculum. *Journal of Systems and Software* 75, 1 (2005), 41–53.
- [11] Kleinner Farias and Bruno C. da Silva. 2020. What's the Grade of Your Diagram? Towards a Streamlined Approach for Grading UML Diagrams. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings (MODELS '20)*. Association for Computing Machinery, 1–2.
- [12] Reza Fauzan, Daniel Siahaan, Siti Rochimah, and Evi Triandini. 2021. Automated Class Diagram Assessment Using Semantic and Structural Similarities. *International Journal of Intelligent Engineering and Systems* 14, 2 (2021), 52–66.
- [13] Sebastian Gürtl, Gloria Schimetta, David Kerschbaumer, Michael Liut, and Alexander Steinmaurer. 2025. Automated Feedback on Student-Generated UML and ER Diagrams Using Large Language Models. arXiv:2507.23470 [cs]
- [14] Robert W. Hasker. 2011. UMLGrader: An Automated Class Diagram Grader. *J. Comput. Sci. Coll.* 27, 1 (2011), 47–54.
- [15] John Hattie and Helen Timperley. 2007. The Power of Feedback. *Review of Educational Research* 77, 1 (2007), 81–112.
- [16] P.B. Henderson. 2003. The Role of Modeling in Software Engineering Education. In *33rd Annual Frontiers in Education, 2003. FIE 2003.*, Vol. 3. S1C–21.
- [17] Gil Hoggarth and Mike Lockyer. 1998. An Automated Student Diagram Assessment System. *SIGCSE Bull.* 30, 3 (1998), 122–124.
- [18] Stephan Krusche. 2022. Semi-Automatic Assessment of Modeling Exercises Using Supervised Machine Learning. In *Hawaii International Conference on System Sciences*. hdl:10125/79439
- [19] Stephan Krusche and Andreas Seitz. 2018. Artemis: An Automatic Assessment Management System for Interactive Learning. In *Proceedings of the 49th Technical Symposium on Computer Science Education (SIGCSE)*. ACM, 284–289.
- [20] Stephan Krusche, Nadine Von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. 2020. An Interactive Learning Method to Engage Students in Modeling. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*. ACM, 12–22.
- [21] Pedro C. Mendonça, Filipe Quintal, and Fábio Mendonça. 2025. Evaluating LLMs for Automated Scoring in Formative Assessments. *Applied Sciences* 15, 5 (2025), 2787.
- [22] Anderson Pinheiro Cavalcanti, Rafael Ferreira Leite de Mello, Vitor Rolim, Máverick André, Fred Freitas, and Dragan Gašević. 2019. An Analysis of the Use of Good Feedback Practices in Online Learning Courses. In *2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, Vol. 2161-377X. 153–157.
- [23] Per Runeson and Martin Höst. 2009. Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164.
- [24] Maximilian Sölch, Felix T. J. Dietrich, and Stephan Krusche. 2025. Direct Automated Feedback Delivery for Student Submissions Based on LLMs. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion '25)*. Association for Computing Machinery, 901–911.
- [25] Dave R. Stikkorum, P. V. D. Putten, Caroline Sperandio, and M. Chaudron. 2019. Towards Automated Grading of UML Class Diagrams with Machine Learning. In *BNAIC/BENELEARN*.
- [26] Michael Striewe and Michael Goedicke. 2014. Automated Assessment of UML Activity Diagrams. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)*. Association for Computing Machinery, 336.
- [27] Pete Thomas, Neil Smith, and Kevin Waugh. 2008. Automatically Assessing Graph-based Diagrams. *Learning, Media and Technology* 33, 3 (2008), 249–267.
- [28] James Vallino. 2007. If You're Not Modeling, You're Just Programming: Modeling Throughout an Undergraduate Software Engineering Program. In *Models in Software Engineering*, Thomas Kühne (Ed.). Springer, 291–300.
- [29] Yen Vo, Heather Rickels, Catherine Welch, and Stephen Dunbar. 2023. Human Scoring versus Automated Scoring for English Learners in a Statewide Evidence-Based Writing Assessment. *Assessing Writing* 56 (2023), 100719.
- [30] Chong Wang, Beian Wang, Peng Liang, and Jie Liang. 2026. Assessing UML Diagrams by GPT: Implications for Education. *Journal of Systems and Software* 234 (2026), 112709.
- [31] Ruiqi Wang, Jiyu Guo, Cuiyun Gao, Guodong Fan, Chun Yong Chong, and Xin Xia. 2025. Can LLMs Replace Human Evaluators? An Empirical Study of LLM-as-a-Judge in Software Engineering. *Proc. ACM Softw. Eng.* 2, ISSTA (2025), ISSTA086:1955–ISSTA086:1977.
- [32] Amali Weerasinghe and Bernard Evans. 2015. UML-IT: An ITS to Teach Multiple Modelling Tasks. In *Artificial Intelligence in Education*, Cristina Conati, Neil Heffernan, Antonija Mitrovic, and M. Felisa Verdejo (Eds.). Springer International Publishing, 816–819.
- [33] Wei Xia, Shaoguang Mao, and Chanjing Zheng. 2024. Empirical Study of Large Language Models as Automated Essay Scoring Tools in English Composition—Taking TOEFL Independent Writing Task for Example. arXiv:2401.03401 [cs]