Towards Automated Evaluation of UML Models in Teaching Software Engineering

Guoda Stulpinaitė [1] and Mantas Jurgelaitis [1]

[1] *Department of Information Systems, Faculty of Informatics, Kaunas University of Technology, Studentų St. 50, Kaunas, 51368, Lithuania*

Abstract

The teaching of Unified Modeling Language (UML) modeling in higher education requires continuous feedback and detailed evaluation of student's models. However, manual evaluation of UML models is time-consuming and requires significant effort from lecturers, particularly when evaluating syntactic correctness, semantic consistency, and vertical and horizontal consistency between models. This paper explores the possibility of an automated approach for providing feedback and preliminary evaluation of UML models in the educational process using rule-based validation and Large Language Models (LLMs). The proposed approach integrates UML metamodel validation, consistency checking, and LLM-based semantic analysis.

Keywords

UML, LLM, model evaluation, horizontal consistency, vertical consistency

1. Introduction

Unified Modeling Language (UML) is a standardized modeling language, a tool that system architects, engineers, and developers can use to analyze, design, and implement information systems [1]. UML is widely taught in study programs related to information systems development because it enables the illustration of core software concepts [2], supports knowledge formalization about systems, and facilitates communication of system requirements and implementation details [3].

In higher education, UML modeling is typically thought across multiple abstraction levels and in various development stages, including requirements, requirements analysis, system design, and implementation modeling. Students are expected not only to produce syntactically correct diagrams but also to ensure semantic validity and consistency between models. This ensures that the requirements for the information system, the concepts of the subject area and the system itself, and the relationships between system components and diagrams are unambiguous. The teaching of UML modeling therefore requires systematic evaluation and meaningful feedback. However, in most academic settings, feedback provision and assessment remain largely manual processes. Students typically receive comments during reviews or evaluation sessions rather than during the modeling process itself. As a result, they may repeat mistakes or fail to identify inconsistencies early. From the lecturer's perspective, evaluating multiple interrelated diagrams while ensuring horizontal and vertical consistency requires significant time and effort.

This calls for more detailed consideration of the use of Large Language Models (LLMs) in teaching UML. LLMs are artificial intelligence systems trained to understand and generate coherent text based on syntactic and semantic patterns [4]. Although LLMs have been explored in software engineering education, their application in UML model evaluation remains limited and often restricted to specific diagram types, such as class diagrams [5]. Moreover, existing approaches do not sufficiently address multilingual modeling contexts or cross-diagram consistency.

CEUR-WS.org/Vol-4317/paper16.pdf

*IVUS 2026: Information Society and University Studies 2026, May 12, Kaunas, Lithuania

guoda.stulpinaite@ktu.edu (G. Stulpinaite); mantas.jurgelaitis@ktu.lt (M. Jurgelaitis)

0000-0003-2221-0765 (M. Jurgelaitis)

Copyright © 2026 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-06

The aim of this research is to explore the challenges and possibilities of applying LLMs to automate UML model evaluation and feedback in education by combining rule-based validation with LLM-driven semantic analysis. The proposed approach seeks to support students during modeling activities while reducing lecturer's workload, without eliminating the necessity of expert supervision.

The rest of the paper is structured as follows. Section 2 discusses the main challenges in providing feedback and evaluating UML models in educational contexts, with particular emphasis on syntactic correctness, semantic validity, and cross-diagram consistency, while also reviewing existing approaches and tools that have been proposed to address these issues. Section 3 presents the proposed UML model evaluation and feedback provision process with an example application. Finally, Section 4 concludes the paper and outlines directions for future research.

2. Background

2.1. Challenges in Providing Feedback and Evaluation of UML Models in Teaching Software Engineering

The evaluation of UML models in educational contexts presents several challenges. UML modeling is inherently multi-layered. In higher education, students construct a requirements model, requirements analysis model, design model, and implementation model. Each model contains multiple diagram types – use case, activity, class, state machine, sequence, package, component, and deployment diagrams – each governed by specific syntactic and semantic constraints.

Another challenge relates to maintaining horizontal consistency. Horizontal consistency refers to the internal consistency of diagrams at the same abstraction level [6]. For example, sequence diagram messages must correspond to operations defined in class diagrams; state transitions must reflect events described in activity diagrams; and actors in use case diagrams must align with interaction participants elsewhere in the model. Violations of such consistency frequently occur in student work.

In addition, vertical consistency introduces additional complexity. Vertical consistency concerns alignment between models at different abstraction levels [6]. Transformations from requirements to design, and from design to implementation models must preserve semantics. For instance, use case behavior specified in activity diagrams in requirements model should be traceable to sequence diagrams and reflected in architectural decisions in design model. However, UML specifications do not provide a unified formal semantic framework for cross-diagram validation [7], making systematic consistency checking difficult.

Another important challenge is the provision of feedback. Manual feedback provision is time intensive. Lecturer must individually review each student's model, identify syntactic violations, semantic mismatches, redundant elements, and inconsistencies across diagrams. Moreover, feedback and evaluation are often conducted simultaneously [8], increasing cognitive load.

Finally, although automated validation mechanisms exist in some Computer Aided Software Engineering (CASE) tools, they are typically limited to rule-based syntactic checks and do not capture higher-level semantic reasoning or pedagogical criteria [9], [10], [11], [12]. Consequently, there is a clear need for automated approaches capable of supporting both formal validation and conceptual feedback.

2.2. Existing UML Model Evaluation and Feedback Solutions

Automated support for UML evaluation traditionally relies on rule-based validation. Many CASE tools – such as Magic Systems of Systems Architect [9], StarUML [10], Modelio [11], and Enterprise Architect [12] – provide built-in validation rules to check diagram correctness. These rules typically verify compliance with UML metamodel constraints and Object Constraint Language (OCL) specifications.

Rule-based approaches are effective in detecting syntactic violations: missing associations, incorrect cardinalities, invalid element types, or structural inconsistencies [13]. However, they are limited in evaluating semantic adequacy. For example, while a tool can verify that a sequence message corresponds to an operation in a class diagram, it cannot determine whether the operation logically fulfills the intended use case scenario.

Recent research explores the integration of artificial intelligence – LLMs – in software modeling education. Some studies demonstrate potential of LLMs in supporting code generation, explanation, and design reasoning [14]. A smaller number of works address UML modeling tasks, particularly class diagram generation from textual descriptions [15].

There also are examples of LLMs used in teaching UML for feedback provision. One of the solutions involves Retrieval-Augmented Generation (RAG)-based LLMs. In this experimental study applying a RAG framework to UML modeling education, a recurring problem was identified: students frequently repeat similar modeling mistakes, while lecturers face difficulties in providing timely and individualized feedback due to large class sizes [5].

To address this issue, a cloud-based solution integrated into Visual Paradigm environment was proposed. The system automatically collects structural information about UML projects – specifically class diagrams – using background plugin. The extracted information then is being stored and when a student or a lecturer submits a query, the RAG framework retrieves relevant contextual information from the repository and augments the prompt provided to the LLM. The LLM then generates a context-aware response, identifying modeling errors and suggesting improvements [5].

As a result, the solution does not address the evaluation of full UML model sets that include requirements, requirements analysis, design, and implementation diagrams even though it is effective within a narrow scope.

Another line of research proposes automated UML evaluation based on comparison with a reference (lecturer-provided) model [8]. In this approach, student diagrams are transformed into XMI format, and structural elements (classes, attributes, relationships, etc.) are extracted and categorized. An algorithm then compares the student's model with the reference solution, identifies differences, generates error list, and computes a grade according to predefines assessment criteria. Experimental results demonstrate that this method enables consistent and scalable automation of UML class diagram grading while reducing lecturer workload [8].

However, this methodology assumes that the quality of student's diagram can be objectively measured based on similarity to a single "correct" solution. Such an assumption may limit modeling creativity and alternate valid design solutions. Moreover, like the RAG-based approach, this solution primarily focuses on class diagrams and does not provide comprehensive support for evaluating multi-layer UIML models. Additionally, it operates mainly as an assessment tool rather than as an assistant for learning UML.

Further development in LLM-based feedback systems is represented by tools such as DUET [16]. Unlike strict reference-model comparison approach, DUET is designed as a learning-oriented assistant. It uses LLMs to generate textual representations of both student and lecturer diagrams and compares them using another LLM configuration. Based on this comparison, structured feedback is generated for both the student and the lecturer [16]. The methodology acknowledges that UML learning involves not only syntactic correctness but also contextual understanding, semantic reasoning, and reflection.

Although DUET extends analysis beyond class diagrams by including sequence diagrams, its scope remains limited in the context of comprehensive system modeling courses that require evaluation across requirements, requirements analysis, and implementation layers.

Overall, existing solutions reveal trade-off between syntactic rigor and semantic flexibility. Rule-based systems ensure structural correctness but lack deep semantic reasoning. LLM-based systems provide richer explanatory feedback and semantic interpretation but often lack strict formal validation guarantees. Furthermore, most approaches concentrate on isolated diagram types rather than addressing horizontal and vertical consistency.

These limitations highlight the need for a methodology that combines rule-based validation with LLM-driven semantic analysis and supports evaluation across multiple abstraction levels withing the educational modeling process.

3. Applying Rule-Based and LLM-Driven Principles in UML Model Evaluation

3.1. The Proposed UML Model Evaluation and Feedback Provision Process

Our proposal focusses on supporting evaluation across multiple modeling stages, including requirements, design, and implementation models, while ensuring both horizontal and vertical consistency and emphasizes the provision of individualized feedback to assist learning and model improvement.

As illustrated in Figure 1, the teaching of UML modeling consists of three primary modeling layers: requirements model (which includes use case, activity, domain class and state machine diagrams), design model (with its sequence, class, package diagrams), and implementation model (including component and deployment diagrams). The actual model evaluation is performed not only within each modeling layer (horizontal consistency) but also across layers (vertical consistency), ensuring consistency throughout the system development lifecycle.

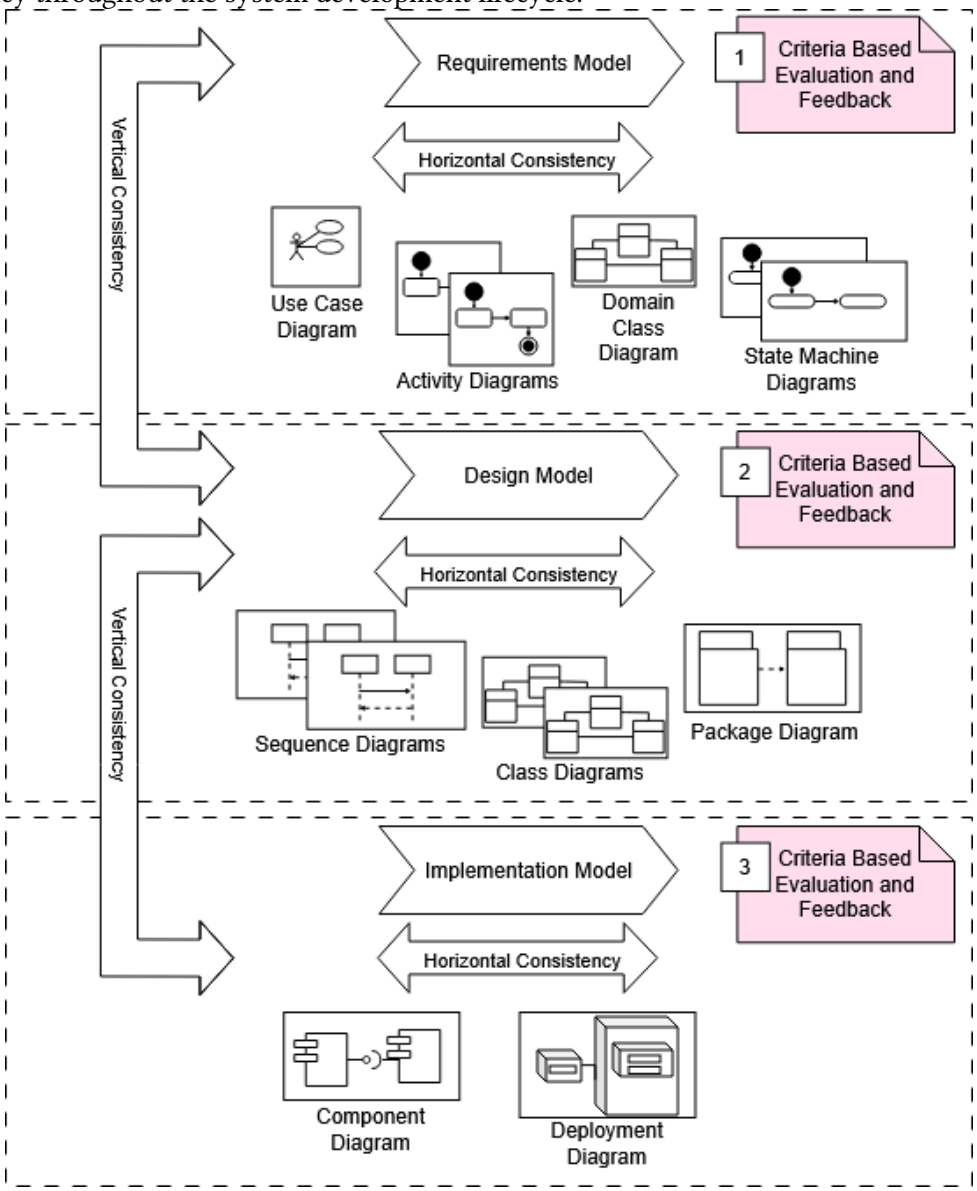


Figure 1: The main concepts of the proposed evaluation process

Rule-based validation and LLMs are intended to support model evaluation by assessing compliance with a defined set of criteria specific to each layer, including the requirements, design, and implementation models.

The evaluation of requirements model focuses on the completeness, correctness, and clarity of the produced diagrams. Criteria assess aspects such as proper structuring of use cases, correct application of relationships (e.g., include, extend, generalization), and the logical consistency of activity flows. Additionally, domain class diagrams are evaluated based on correct use of relationships, attributes, and stereotypes, while state machine diagrams must accurately capture the behavior of selected class. A key consideration is that within the requirements model, each use case identified during the construction of the use case diagram should be complemented by a corresponding activity diagram and a sequence diagram, the latter typically belonging to the design model. Furthermore, in the domain class diagram, all classes that require behavioral specification should be accompanied by an associated state machine diagram.

At the design model layer, the evaluation emphasizes the transformation of requirements into a system-level design. Criteria focus on sequence diagrams, design-level class diagrams, and package diagrams representing the system's logical architecture. Particular attention is given to the consistency between diagrams: messages in sequence diagrams must correspond to operations in class diagrams, and design elements should clearly trace back to requirements model artifacts. It is important to distinguish the difference between class diagrams developed in the design model and those in the requirements model as design model class diagrams represent system-level constructs, such as boundary, controller, and model classes, whereas domain class diagram captures conceptual entities of the problem domain.

The evaluation in implementation layer consists of assessment of component and deployment diagrams in terms of their correspondence to the system architecture defined in the design model, ensuring that the identified components, their relationships, and deployment configurations reflect the intended system structure.

For the evaluation and feedback provision process both validation rules and LLMs are employed. Validation rule-based evaluation is derived from UML metamodel constraints, and course-specific assessment criteria. This ensures that modeling errors are detected automatically. However, rule-based mechanisms remain limited to structural and syntactic dimensions, they cannot evaluate the appropriateness of modeling decisions. LLMs are applied to complement rule-based validation by providing semantic interpretation and qualitative feedback. The LLM-driven component analyzes textual representations of UML diagrams, evaluates naming conventions and conceptual clarity, assesses logical reasoning behind modeling decisions, and generates structured feedback aimed at improving model quality. However, LLM outputs are treated as advisory and lecturer's oversight remains a required component of the final evaluation process.

From a pedagogical perspective, the proposed automated process supports early-stage feedback during modeling activities, reducing repeated mistakes and promoting iterative refinement. Lecturers benefit from reduced repetitive evaluation tasks and have more time for discussions.

Figure 2 illustrates the proposed concept of model evaluation and feedback provision workflow. The process consists of several sequential steps that combine rule-based validation with subsequent LLM-driven evaluation procedures. During the model validation step (A1), validation rules are applied from a predefined rule set, after which the resulting validation outputs are presented and can be exported. Furthermore, during the model analysis step (A6), both the model and its associated validation results are examined. At this step, LLMs are employed to interpret the model according to the specified evaluation criteria. Subsequently, in the evaluation step (A7), structured feedback is generated for the model, accompanied by a preliminary grade assigned to the student. The final grading decision, however, remains the responsibility of the lecturer.

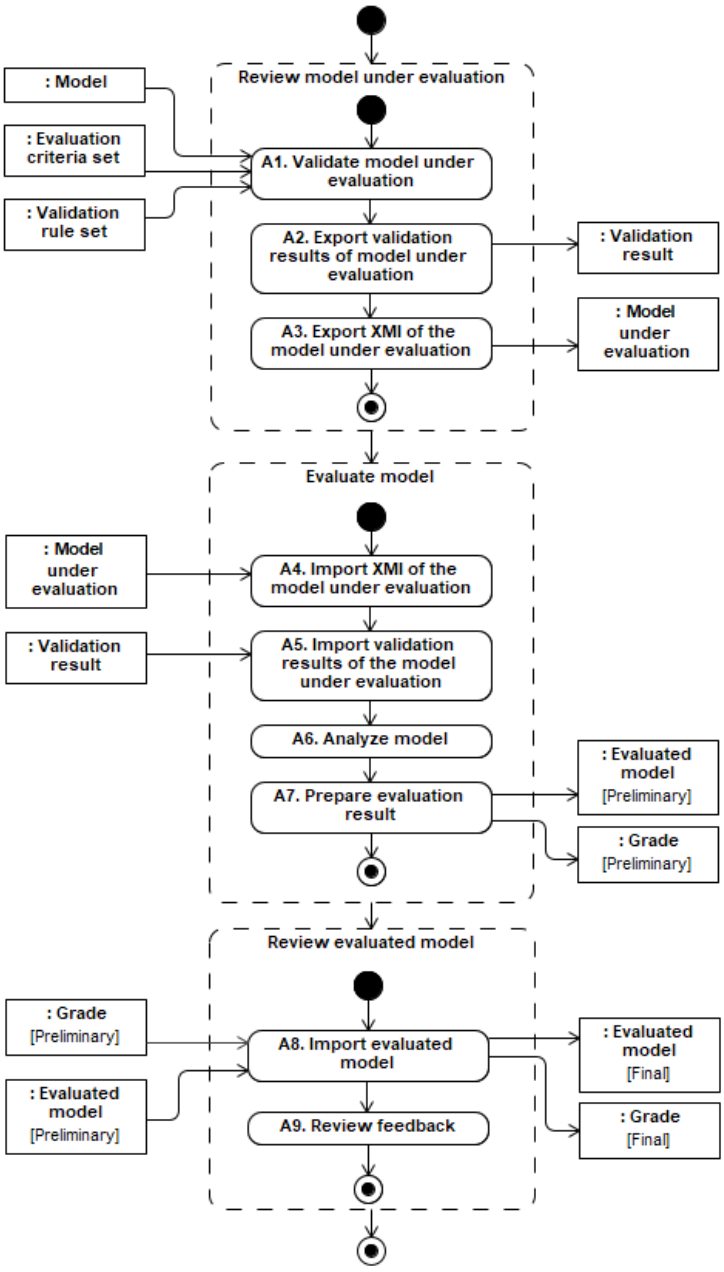


Figure 2: General process of the UML model evaluation and feedback provision methodology

3.2. Example Application of Rule-Based and LLM-Driven UML Model Evaluation

To demonstrate the proposed evaluation approach, we present an illustrative example of a Bookstore information system UML model. The purpose of this example is to show how the combination of rule-based validation and LLM-driven analysis can be used in practice to evaluate and provide feedback on student's model. The fragment of Bookstore requirements model is presented in Figure 3 encompassing use case, activity and domain class diagrams.

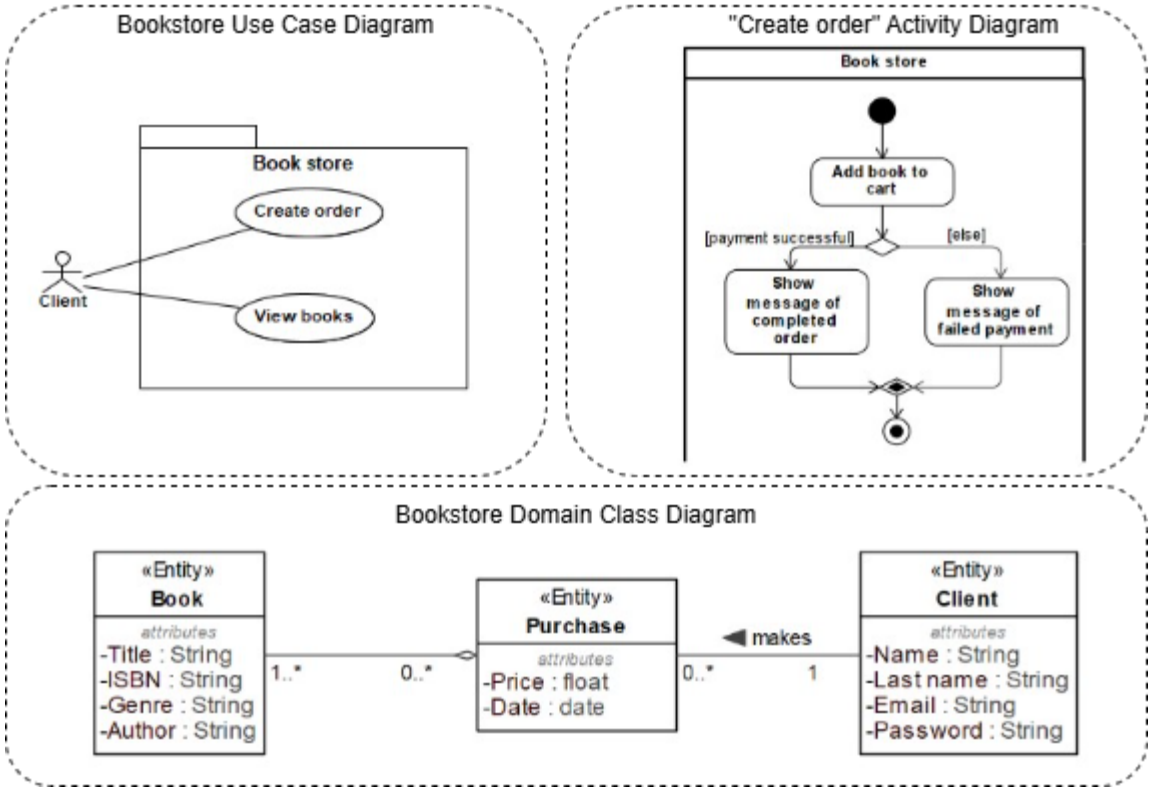


Figure 3: Bookstore information system requirements model fragment

One of the requirements model evaluation criteria requires that activity diagrams are directly associated to the corresponding use cases and include at least two swimlanes representing the user and the system (except for system-level use cases involving only the system). In Figure 3, the use case "Create order" is performed by a client and is detailed in an associated activity diagram. A rule-based validation mechanism can be employed to automatically verify whether the activity diagram satisfies the specified criteria, including the presence of the required swimlanes and the consistency of actors with those defined in the corresponding use case. Such a rule may be formalized using a constraint language such as the Object Constraint Language (OCL), as illustrated in Figure 4.

```
context UseCase
inv SwimlanesMatchActorsPlusOne:
let relatedActivity : Activity =
    self.ownedBehavior->select(oclIsKindOf(Activity))->any(true).oclAsType(Activity)
in
relatedActivity <> null implies
    relatedActivity.partition->size() = self.subject->collect(a | a)->size() + 1
```

Figure 4: OCL rule example for the correct number of activity diagram swimlanes

From a semantic and horizontal consistency perspective, the domain class diagram in Figure 3 contains a discrepancy: one of the classes, "Purchase", is inconsistent with the terminology used in the use case and activity diagrams, where the equivalent concept is denoted as "Order". This inconsistency indicates a lack of alignment across models and may lead to ambiguity in the system representation.

To address this issue, LLMs may be used to assess whether each class name is referenced in at least one use case name within the use case diagram and/or in action labels within the corresponding activity diagrams. In cases where no direct correspondence is identified, LLMs can be utilized to detect potential semantic equivalence by identifying synonyms or closely related terms, thereby supporting the resolution of naming inconsistencies across models.

Once such model inaccuracies are identified explanatory feedback for students can be incorporated into the model. The feedback comments can be linked to the specific model element in which the issue was detected, thereby enabling students to locate and address the problem. Examples of such feedback, attached to model elements, are presented in Figure 5 and Figure 6.

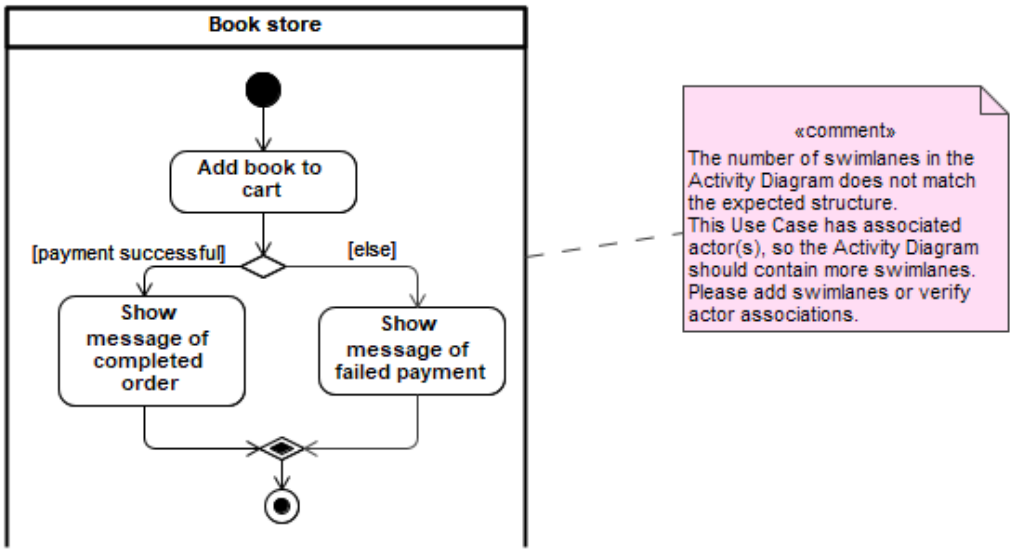


Figure 5: Example of OCL validation rule-based validation related feedback comment

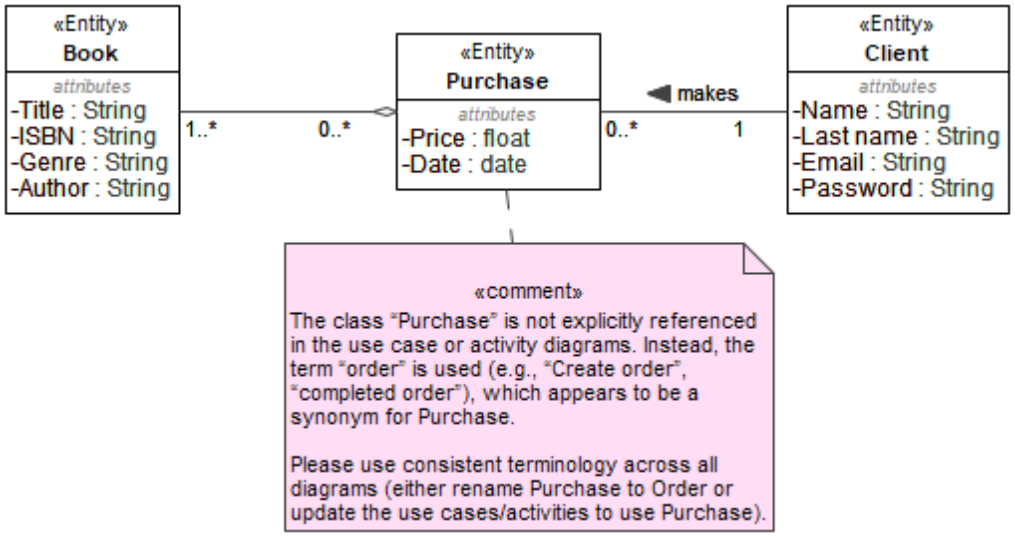


Figure 6: Example of LLM-driven analysis related feedback comment

This example demonstrates how the combined application of rule-based validation and LLM-driven analysis facilitates the detection of structural and semantic inconsistencies in UML models, while enabling the generation of context-aware feedback.

4. Conclusion

UML modeling education requires systematic evaluation of syntactic correctness, semantic validity, and cross-diagram consistency. Manual feedback provision and evaluation processes are time-consuming and cognitively demanding, particularly when multiple abstraction layers must be evaluated simultaneously.

We propose a concept of an automated process for UML model evaluation in educational contexts where the UML model under evaluation is exported in XMI format, along with validation results based on OCL constraints. These artifacts are subsequently analyzed using a combination of rule-

based techniques and LLM-driven methods. Rule-based mechanisms ensure formal compliance with UML metamodel constraints, while LLMs provide higher-level semantic insights and explanatory feedback. The proposed approach does not aim to fully replace lecturer judgement. Instead, it supports formative learning processes, enhances feedback timeliness, and reduces repetitive validation tasks. In the future we are planning to focus on implementing the proposed evaluation process as a specialized tool and experimentally evaluating its effectiveness in real educational settings.

5. Declaration on Generative AI

The authors have not employed any Generative AI tools.

6. References

[1] O. M. Group, "OMG Unified Modeling Language (OMG UML), Version 2.5.1," Object Management Group, 2017.

[2] G. Engels and e. al., "Teaching UML Is Teaching Software Engineering Is Teaching Abstraction," in *Satellite Events at the MoDELS 2005 Conference*, Springer, 2006, p. 306–319.

[3] S. S. Alhir, Understanding the Unified Modeling Language (UML), Prentice Hall, 1998.

[4] E. D. P. Supervisor, "Large Language Models (LLM)," 2025. [Online]. Available: https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en.

[5] P. Ardimento and e. al., "Teaching UML Using a RAG-Based LLM," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024.

[6] M. Elaasar and L. Briand, "An Overview of UML Consistency Management," 2016.

[7] A. Knapp and T. Mossakowski, "Multi-view Consistency in UML," 2002.

[8] R. Jebli and e. al., "A Proposed Algorithm for Assessing and Grading Automatically Student UML Diagrams," *International Journal of Modern Education and Computer Science,* vol. 16, no. 1, p. 37, 2024.

[9] N. M. Inc., "Magic Systems of Systems Architect Documentation," 2024. [Online]. Available: https://docs.nomagic.com/spaces/MSOSA190SP3/pages/54168292/Magic+Systems+of+Systems+Architect+Home+Page.

[10] StarUML, "StarUML User Interface Documentation," 2025. [Online]. Available: https://docs.staruml.io/user-guide/user-interface.

[11] M. O. Source, "Modelio User Documentation," 2025. [Online]. Available: https://github.com/ModelioOpenSource/Modelio/wiki/Modelio-User-Documentation.

[12] S. Systems, "Enterprise Architect," 2015. [Online]. Available: https://sparxsystems.com/products/ea/12/.

[13] C. M. Zapata and e. al., "A Rule-Based System for Assessing Consistency Between UML Models," in *MICAI 2007: Advances in Artificial Intelligence*, 2007.

[14] E. Kasneci and e. al., "ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education," *Learning and Individual Differences,* vol. 103, 2023.

[15] B. Wang and e. al., "How LLMs Aid in UML Modeling: An Exploratory Study with Novice Analysts," 2024. [Online]. Available: https://arxiv.org/abs/2404.17739.

[16] S. Gürtl and e. al., "Automated Feedback on Student-Generated UML and ER Diagrams Using Large Language Models," 2025. [Online]. Available: https://arxiv.org/abs/2507.23470.