

LLM-UML Benchmark: A Reproducible Benchmark for Evaluating Large Language Models on UML Diagram Interpretation

Hamdan Alabsi
halabsi@bowiestate.edu
Bowie State University
Bowie, MD 20715, USA

Sriram Srinivasan
ssrinivasan@bowiestate.edu
Bowie State University
Bowie, MD 20715, USA

Majed Almotairi
malmotairi@uncfsu.edu
Fayetteville State University
Fayetteville, NC 28301, USA

Mohammed Alyami
malyami@coloradomesa.edu
Colorado Mesa University
Grand Junction, CO 81501, USA

Rand Obeidat
robeidat@bowiestate.edu
Bowie State University
Bowie, MD 20715, USA

Azene Zenebe
azenebe@bowiestate.edu
Bowie State University
Bowie, MD 20715, USA

Abstract

Despite growing interest in applying multimodal large language models (LLMs) to Systems Analysis and Design (SAD), researchers lack standardized and reproducible methods for evaluating UML diagram interpretation. We present LLM-UML Benchmark v1.0, a reproducible benchmark for evaluating diagram-to-structured UML interpretation under a common experimental protocol. The benchmark comprises 90 cases created by one subject-matter expert (SME), including 30 use case, 30 class, and 30 sequence diagrams evenly distributed across low-, medium-, and high-complexity levels. Two additional SMEs independently evaluated the rendered diagrams using a standardized rubric covering notation correctness, readability, coherence, complexity fit, and answer-key interpretability. Each case includes a canonical PlantUML specification, rendered diagram, benchmark metadata, creator-developed canonical JSON reference, diagram-specific JSON schema, standardized prompt, and evaluation metadata. We further introduce the UML Understanding Assessment Framework (UUAF), which evaluates structural, relationship, behavioral, functional, and representation quality to produce case-level and aggregate model scores. Using a standardized evaluation pipeline, we benchmarked GPT-4.1, Claude Sonnet 5, and Gemini 3.1 Pro Preview. Gemini achieved the highest descriptive overall score (0.801), followed by Claude (0.780) and GPT-4.1 (0.764). Although overall performance differences were not statistically significant, meaningful variation emerged across diagram types and evaluation dimensions: GPT-4.1 performed best on use case diagrams, Gemini on class diagrams, and Claude on sequence diagrams. Across all models, explicit UML structures were interpreted more reliably than benchmark-aligned functional requirements. LLM-UML Benchmark provides a standardized, extensible, and reproducible foundation for evaluating multimodal LLMs and supports future research, instructional evaluation, and AI-assisted systems analysis and design.

Keywords: large language models, UML, benchmark, systems analysis and design, multimodal AI, reproducibility

LLM-UML Benchmark: A Reproducible Benchmark for Evaluating Large Language Models on UML Diagram Interpretation

*Hamdan Alabsi, Sriram Srinivasan, Majed Almotairi,
Mohammed Alyami, Rand Obeidat and Azene Zenebe*

1. INTRODUCTION

Unified Modeling Language (UML) diagrams remain central to systems analysis, software design, and information systems education because they externalize system structure and behavior through standardized visual notations. Researchers have repeatedly shown that diagram quality, notation choice, and modeling expertise affect comprehension, defect detection, and communication (Agarwal et al., 1999; Gemino & Wand, 2004; Lange & Chaudron, 2006; Moody, 2009; Petre, 2013). Instructors therefore use use case, class, and sequence diagrams to develop students' ability to connect stakeholder goals, domain structure, interactions, and functional requirements (Daun et al., 2017; Siau & Loo, 2006).

Multimodal large language models now create a new opportunity and a new evaluation problem. These models can receive diagram images and produce explanations, structured representations, requirements, or code. Research on language models for software engineering has expanded rapidly across code generation, maintenance, testing, and design-oriented tasks (Arpteg et al., 2018; Chen et al., 2021; Hou et al., 2024; Jimenez et al., 2024; Lu et al., 2021; Tufano et al., 2019). However, existing software-engineering benchmarks emphasize source code, issue resolution, or textual reasoning. They do not provide a common dataset, prompt protocol, canonical output representation, and diagram-specific scoring framework for UML interpretation. Consequently, model comparisons can conflate genuine capability differences with differences in prompts, schemas, normalization, gold standards, and scoring rules.

We addressed this gap by developing LLM-UML Benchmark v1.0. We designed the benchmark as a specification-centered research artifact: the benchmark defines tasks, cases, schemas, prompts, gold standards, evaluation rules, and reproducibility requirements, while the code implements those specifications. This design follows established lessons from benchmark-centered evaluation in natural language understanding and software engineering, where standardized cases and transparent metrics enable cumulative comparison (Hendrycks et al., 2021; Jimenez et al., 2024; Lu et al., 2021). These principles are also consistent with evidence-based software engineering practices that emphasize systematic, reproducible empirical evaluation (Budgen et al., 2011).

We organized the study around six research questions aligned with the UML Understanding Assessment Framework (UUAF):

RQ1: How accurately do multimodal LLMs identify structural UML elements?

RQ2: How accurately do multimodal LLMs preserve semantic relationships among UML elements?

RQ3: How accurately do multimodal LLMs interpret behavioral semantics in Sequence diagrams?

RQ4: How accurately do multimodal LLMs infer functional requirements that are explicitly represented or directly supported by UML diagrams?

RQ5: How reliably do multimodal LLMs generate machine-processable structured UML representations suitable for automated evaluation?

RQ6: How does UML interpretation performance vary across models, diagram types, and benchmark complexity levels?

Our main contributions are: (1) a reproducible benchmark containing 90 controlled UML cases across three diagram types and three complexity levels; (2) canonical, diagram-specific schemas and prompts that impose a common output contract across model providers; (3) a multi-dimensional UML Understanding Assessment Framework that separates structural, relationship, behavioral, functional, and representation quality; (4) an empirical validation using 270 paired model-diagram observations

and paired nonparametric statistical tests; and (5) an open research-software architecture that supports model extension, automated evaluation, and future benchmark tasks.

2. RELATED WORK

Conceptual-modeling research provides the methodological foundation for evaluating UML interpretation. Early frameworks distinguished syntactic, semantic, and pragmatic model quality and emphasized correspondence between a model, its domain, and its audience (Lindland et al., 1994; Wand & Weber, 2002). Later work developed quality frameworks, diagram metrics, and empirical evaluation approaches for model-based development (Genero et al., 2005; Gemino & Wand, 2004; Krogstie et al., 2006; Mohagheghi et al., 2009; Siau & Rossi, 2011). Studies of UML usage found that practitioners apply the notation selectively, value diagrams for communication, and frequently adapt formal conventions to project needs (Dobing & Parsons, 2006; Nugroho & Chaudron, 2008; Petre, 2013; Storrie, 2017). These findings motivated our decision to score explicit elements and relationships separately rather than treating a diagram as a single undifferentiated artifact.

Researchers have also documented UML-specific comprehension challenges. Defects and inconsistent conventions can impair comprehension and maintenance decisions (Lange & Chaudron, 2006; Lange et al., 2006). Visual syntax, mandatory properties, and notation complexity influence how readers interpret diagrams (Gemino & Wand, 2005; Moody, 2009; Purchase et al., 2001). Learners particularly struggle to connect static and dynamic views and to translate stakeholder narratives into correct abstractions (Daun et al., 2017; Siau & Loo, 2006). These results support diagram-specific evaluation: actors and use cases do not represent the same reasoning demands as classes, multiplicities, messages, and interaction fragments.

Model-driven engineering research further stresses traceability and consistency across artifacts (Briand et al., 2003; Cabot & Gogolla, 2012; France & Rumpe, 2007; Hutchinson et al., 2011; Mellor et al., 2003). Requirements representations also need to expose ambiguity and incomplete assumptions before implementation (Glinz, 2000). We therefore included functional understanding as a separate dimension: a model may reproduce visible UML elements while still failing to infer a complete and supported set of requirements.

Modern AI research has produced standardized benchmarks for language, reasoning, code, and software maintenance. MMLU demonstrated the value of broad, controlled task collections (Hendrycks et al., 2021), while CodeXGLUE and related datasets standardized code understanding and generation (Chen et al., 2021; Lu et al., 2021). SWE-bench moved evaluation toward repository-level issue resolution (Jimenez et al., 2024), and SpreadsheetBench extended benchmark design to structured business artifacts (Ma et al., 2024). Software-engineering studies additionally show that code and technical artifacts possess statistical regularities that machine-learning systems can exploit (Hindle et al., 2012; Meyer, 2022; Tufano et al., 2019).

The rapid adoption of generative AI in computing education has intensified the need for task-valid evaluation. Studies have examined prompt design, programming assistance, feedback, and human oversight (Amershi et al., 2014; Denny et al., 2023; Yadav et al., 2016). Cognitive-load research also predicts that well-designed visual and textual representations can support learning, whereas poorly aligned representations increase extraneous processing (Paas et al., 2003; Sweller, 1988). To the best of our knowledge, existing peer-reviewed benchmarks do not jointly integrate UML diagram images, canonical structured outputs, diagram-specific normalization, and multidimensional evaluation. LLM-UML Benchmark addresses this gap by providing a standardized, reproducible evaluation framework that bridges software engineering benchmark practices and Systems Analysis and Design education.

Broader software engineering research reinforces the need for transparent artifact quality and rigorous evaluation. Researchers have examined technical debt, readability, lifecycle risk, and model evolution (Bavota & Russo, 2016; Boehm, 1988; Briand et al., 2003; Buse & Weimer, 2010). Other studies have compared conceptual modeling methods and process representations (Agarwal & Sinha, 2003; Recker et al., 2009; Sedraky & Snoeck, 2016), while reasoning benchmarks and AI ethics research emphasize task validity, reproducibility, and diagnostic reporting (Clark et al., 2018; Zhuo et al., 2023). These findings support our emphasis on traceable gold standards, standardized outputs, and multidimensional evaluation.

Recent work now directly examines multimodal models and software-modeling artifacts. Cámara et al. (2024) proposed a standardized framework for benchmarking LLMs on software-modeling tasks, while Doris et al. (2025) introduced DesignQA for multimodal reasoning over engineering requirements and drawings. UML-focused studies have evaluated semantic-element recognition in use-case diagram images (Hassine, 2025), diagram-image-to-UML-code generation (Bates et al., 2025), and multimodal assessment of student class diagrams (Ibáñez et al., 2025). These studies confirm rapid progress but generally focus on one notation, one transformation, or one assessment context. They do not provide aligned Use Case, Class, and Sequence views of common scenarios together with provider-neutral prompts, canonical structured outputs, deterministic normalization, and a multidimensional interpretation framework. LLM-UML Benchmark addresses this combined gap.

3. LLM-UML BENCHMARK AND METHODS

3.1 Benchmark Scope and Dataset

We designed LLM-UML Benchmark v1.0 as a controlled benchmark for evaluating multimodal large language models on Task 1: Diagram-to-Structured-UML. In this task, an LLM receives a UML diagram as input and interprets its graphical elements to generate a standardized JSON representation that preserves the diagram's semantic meaning. In addition to extracting explicit UML elements, Task 1 requires models to generate concise, high-level functional requirements that can be directly inferred from the diagram. These requirements serve as a bridge between UML interpretation and downstream software analysis but are limited to functionality explicitly represented in the UML model. They are not intended to constitute a complete software requirements specification, which is addressed separately in Task 2: Diagram-to-Functional Requirements Engineering.

Accordingly, Task 1 evaluates UML interpretation by extracting only the functional semantics explicitly represented or directly inferable from the diagram, whereas Task 2 evaluates broader software requirements engineering by generating comprehensive functional and nonfunctional requirements, business rules, acceptance criteria, traceability links, and cross-diagram requirements. Depending on the UML diagram type, the extracted information includes actors, use cases, classes, attributes, operations, relationships, participants, messages, interaction sequences, and diagram-supported functional requirements. We developed a canonical JSON schema for each supported UML diagram type to ensure that every model produces a consistent structured representation regardless of provider. This standardized representation enables direct comparison between model outputs and creator-developed canonical reference using objective evaluation metrics, including precision, recall, F1 score, and the UML Understanding Assessment Framework (UUAF).

We designed LLM-UML Benchmark as a versioned and extensible benchmarking framework that supports continuous expansion in both benchmark tasks and UML diagram types. Version 1.0 focuses on the three UML diagram types most frequently taught in Systems Analysis and Design curricula and widely used in software engineering practice: Use Case, Class, and Sequence diagrams. These diagram types capture complementary structural and behavioral perspectives of software systems while representing fundamental learning outcomes in undergraduate Information Systems education. Future benchmark releases will expand the framework to additional UML diagram types, including Activity, State Machine, Component, Deployment, Communication, Package, Composite Structure, Object, Timing, and Interaction Overview diagrams.

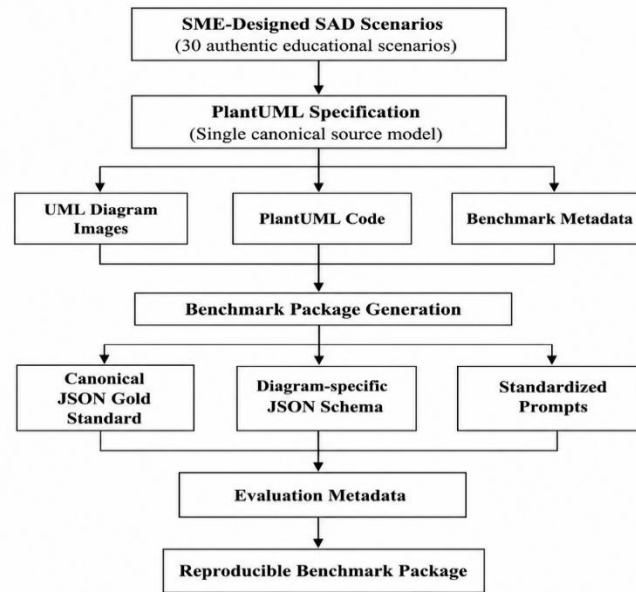


Figure 1. LLM-UML Benchmark construction workflow, illustrating the transformation of expert-designed PlantUML scenarios into a complete, reproducible benchmark package with synchronized benchmark artifacts and evaluation resources.

Figure 1 uses the following artifact definitions. The PlantUML specification is the canonical plain-text source from which the diagram image is rendered. Benchmark metadata record the case identifier, scenario, domain, diagram type, complexity level, and artifact paths. The canonical JSON reference contains the creator-developed expected structured content used for deterministic evaluation. The diagram-specific JSON schema is the machine-readable contract that defines permitted fields, data types, and required fields. The standardized prompt is the provider-neutral instruction sent with the diagram image. Evaluation metadata record the normalization rules, metric mappings, evaluator version, and provenance needed to reproduce each score.

Beyond Task 1, the benchmark specification defines five additional tasks that progressively increase UML interpretation and reasoning complexity: (2) Diagram-to-Functional Requirements Engineering, which extends Task 1 by generating comprehensive software requirements, including refined functional and nonfunctional requirements, business rules, acceptance criteria, traceability links, and cross-diagram requirements; (3) Diagram-to-PlantUML, which reconstructs editable PlantUML source code from UML diagrams; (4) Diagram-to-Natural Language Explanation, which generates human-readable explanations of UML diagrams; (5) Diagram Question Answering, which answers reasoning questions about a single UML diagram; and (6) Cross-Diagram Reasoning, which integrates information across multiple UML diagrams. Together, these tasks provide an extensible roadmap for evaluating multimodal LLMs across multiple levels of UML understanding. The present study evaluates only Task 1 using 90 diagrams spanning the three supported UML diagram types; the remaining tasks are planned for future benchmark releases.

A primary contribution of LLM-UML Benchmark is its SME-developed dataset, which avoids relying on UML diagrams collected from online repositories or automatically generated from examples of uncertain quality. One SME created 30 scenarios and their corresponding 90 UML diagrams—30 use case, 30 class, and 30 sequence diagrams—along with the associated PlantUML specifications, functional requirements, and canonical JSON references. Scenario development was informed by recurring modeling patterns in widely adopted Systems Analysis and Design textbooks (Dennis et al., 2020; Satzinger et al., 2015; Sommerville, 2016) and by broad application themes identified in public GitHub repositories associated with the “srs-document” and “software-requirement-specification” topics (GitHub, 2026a, 2026b). These topic collections reflected common software-system contexts, including healthcare, education, banking, e-commerce, hotel management, transportation, inventory management, and service systems. We consulted these repositories only to identify recurring domains and scenario themes; we did not copy

their requirements, diagrams, source code, or other artifacts. Figure 2 presents examples of three UML diagram types: (a) Use Case, (b) Class, and (c) Sequence.

We selected the scenarios according to four criteria: relevance to Systems Analysis and Design, ability to support coherent use case, class, and sequence views, coverage of multiple application domains, and suitability for controlled comparisons across low, medium, and high complexity levels. Accordingly, the benchmark comprises controlled educational scenarios informed by established textbook patterns and publicly observable software-requirements themes rather than models extracted from proprietary industrial systems. As illustrated in Figure 1, we transformed each scenario into a synchronized benchmark package containing the canonical PlantUML specification, rendered UML diagram, benchmark metadata, canonical JSON reference, diagram-specific JSON schema, standardized prompt, and evaluation metadata. This integrated workflow provides end-to-end traceability from scenario design through benchmark evaluation and supports the reproducibility and reuse of every case.

Two additional SMEs independently evaluated each rendered diagram using a standardized 12-criterion rubric covering complexity-level match, layout readability, text and label legibility, arrow and line clarity, logical relationship correctness, UML notation correctness, visual clutter, spacing and alignment, content coherence, educational suitability, answer-key interpretability, and overall diagram quality. The reviewers also documented observed issues and recommended whether each diagram should be retained, revised, or removed. Their review assessed diagram quality and whether an expert could derive an unambiguous answer key from the rendered diagram; it did not involve independently reconstructing every canonical UML element or functional requirement. Appendix E presents the complete SME evaluation analysis.

Throughout this paper, *canonical reference* denotes the benchmark creator-developed expected structured content used for deterministic evaluation. To reduce subjectivity in Functional Understanding, the benchmark creator expressed each reference requirement as a single atomic system behavior and linked it to explicit UML evidence, such as an actor–use case association, class operation, or message sequence. Requirements dependent on unstated business rules, nonfunctional assumptions, or domain knowledge not represented in the diagram were excluded. The two reviewing SMEs assessed the interpretability of these answer keys through the Gold-Standard Interpretability criterion. This procedure supports the traceability and interpretability of the canonical references but does not constitute independent element-by-element consensus validation.

Because each of the 30 scenarios is represented by all three UML diagram types, the benchmark provides aligned structural and behavioral views of the same software problem. This multi-artifact design supports the present evaluation of multimodal LLMs using UML images while enabling future tasks involving PlantUML reconstruction, functional-requirements generation, natural-language explanation, question answering, and cross-diagram reasoning. The released benchmark includes PlantUML source files, rendered UML images, canonical JSON references, diagram-specific schemas, prompts, metadata, and evaluation tools. These resources provide a common infrastructure for standardized evaluation, reproducibility, and longitudinal comparison of future multimodal LLMs. The benchmark specification, evaluation framework, canonical prompts and schemas, and reference implementation are publicly available in the accompanying GitHub repository. Appendix A summarizes the complete workflow used to reproduce the benchmark results.

To broaden domain coverage, we designed scenarios spanning healthcare, education, e-commerce, service portals, finance, organizational systems, inventory management, library systems, transportation, and customer relationship management. We balanced the benchmark across three complexity levels by developing 10 low-complexity, 10 medium-complexity, and 10 high-complexity diagrams for each UML diagram type. We operationalized complexity using both the number of UML modeling elements and the richness of their structural and behavioral interactions. Low-complexity diagrams contain relatively few modeling elements connected through straightforward relationships. Medium-complexity diagrams introduce additional entities, associations, attributes, alternative interaction paths, or behavioral constructs. High-complexity diagrams combine larger structural models with richer relationships, nested interactions, and more sophisticated behavioral patterns requiring deeper semantic interpretation. This design enables systematic evaluation across increasing levels of modeling complexity while maintaining balanced case counts across UML diagram types. Appendix D presents the diagram-specific operational criteria used to assign complexity levels.

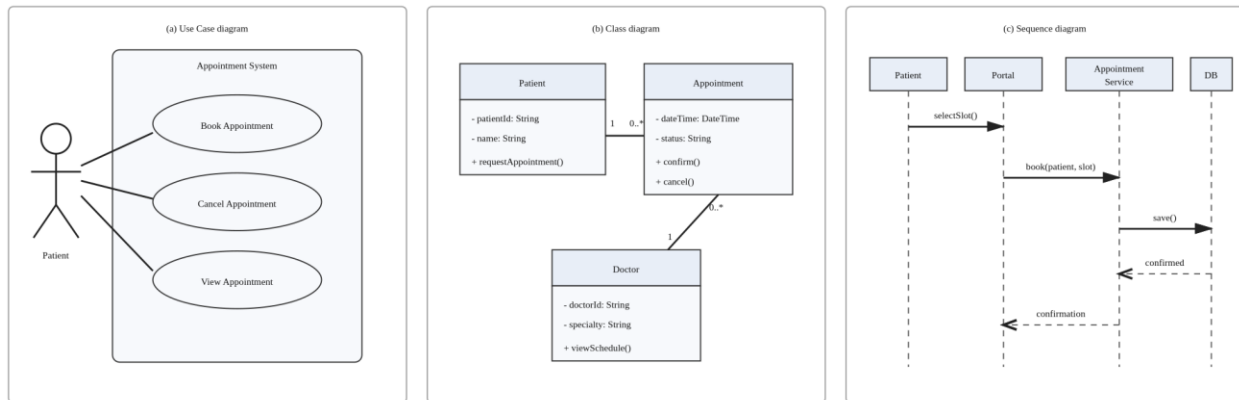


Figure 2. Representative benchmark-format examples for the three evaluated UML diagram types: (a) Use Case, (b) Class, and (c) Sequence. The examples use a common appointment-scheduling domain to illustrate the notation-specific information supplied to each model.

3.2 Prompts, Schemas, Models, and Execution

For each UML diagram type, we created a provider-neutral prompt with fixed instructions, explicit field definitions, diagram-specific extraction rules, and the constraint to return JSON only. We use “tightly structured prompt” to refer to this fixed instruction template. The corresponding canonical schema is a JSON Schema that specifies the allowed object structure, field names, data types, required fields, and controlled relationship categories. The same prompt–schema pair was used across all providers. Use Case outputs captured actors, use cases, associations, include and extend relationships, generalizations, and functional requirements. Class outputs captured classes, attributes, operations, typed relationships, multiplicities, and functional requirements. Sequence outputs captured participants, ordered messages, message types, combined fragments, and functional requirements. Figures 3 and 4 illustrate the prompt–schema workflow and an annotated Sequence-diagram schema excerpt, respectively.

We selected three commercial multimodal models (OpenAI gpt-4.1, Anthropic claude-sonnet-5, and Google gemini-3.1-pro-preview) because, at the July 14, 2026 execution date, each offered image input, API access, and structured-output capability suitable for the same diagram-to-JSON protocol. Selecting one model from each of three major providers also reduced dependence on a single vendor architecture. We used the exact API identifiers recorded in the run manifests and ran each model on all 90 diagrams, yielding 270 paired model–diagram observations. The benchmark is not limited to these providers: any commercial or open-weight vision-language model that accepts an image and can return the schema fields can be evaluated through an adapter. Future evaluations will include vision-capable Qwen, Llama, and DeepSeek-family checkpoints, subject to model availability and reproducible deployment settings. The repository preserved run identifiers, timestamps, provider metadata, decoding parameters, raw outputs, parsed outputs, execution manifests, and evaluation artifacts for every experiment. Table 1 summarizes the configuration.

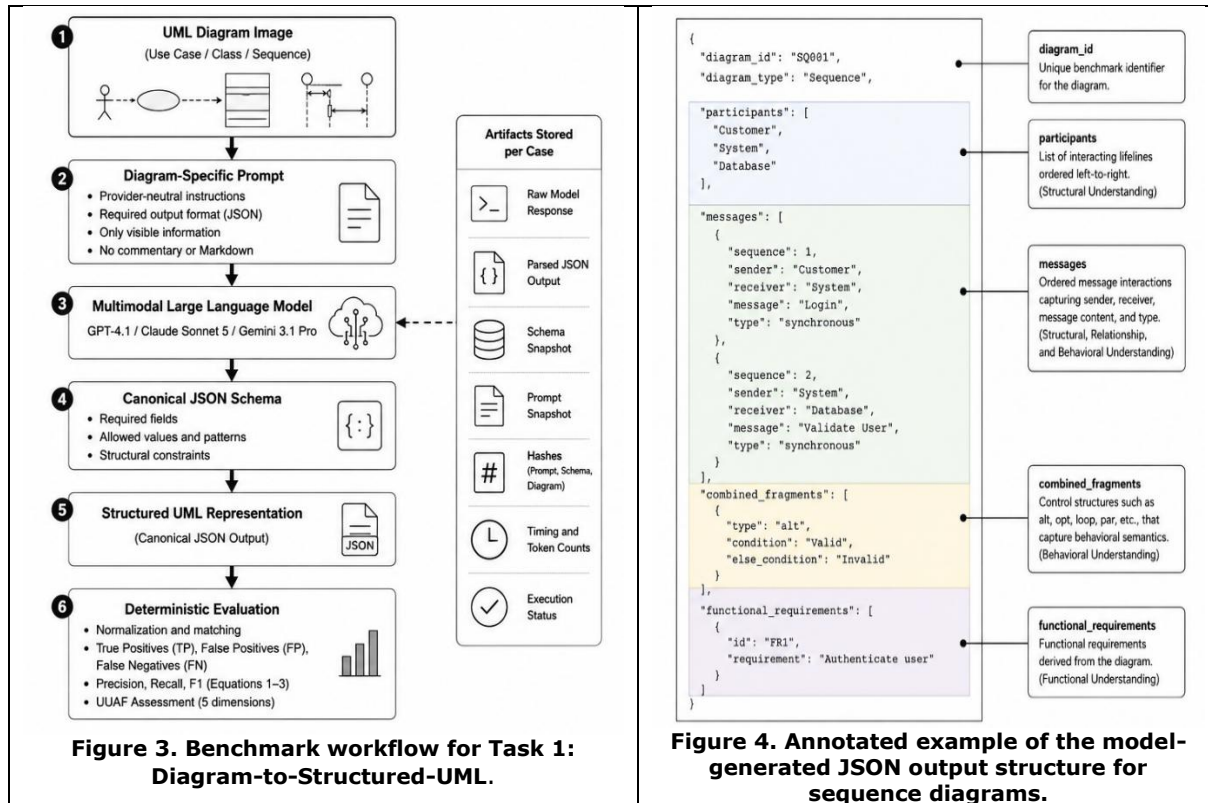
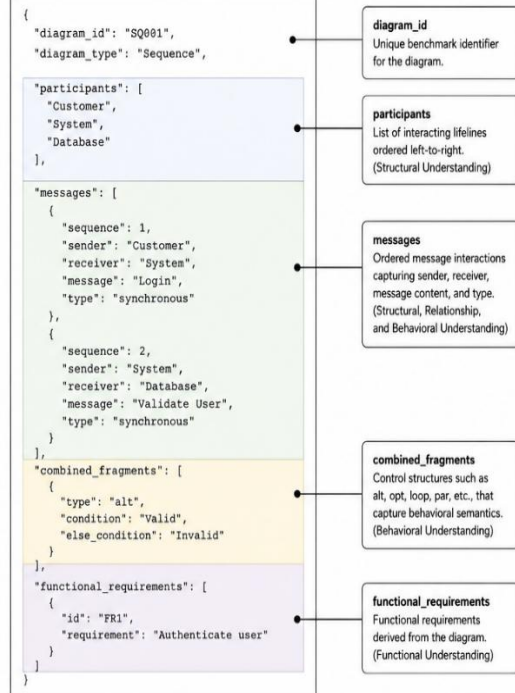


Figure 4. Annotated example of the model-generated JSON output structure for sequence diagrams.



Before comparison, we normalized strings, collections, identifiers, relationship direction, multiplicities, and diagram-specific fields. “Canonical-reference UML elements” denotes the expected actors, use cases, classes, attributes, operations, participants, messages, relationships, and functional requirements authored by the benchmark creator and recorded in the canonical JSON reference for each diagram. The evaluator matched predicted elements against these expected elements and computed true positives (TP), false positives (FP), and false negatives (FN). We calculated precision, recall, and F1 using Equations (1)–(3).

Component	Configuration
Benchmark task	Diagram-to-Structured-UML (Task 1)
Benchmark cases	90 UML diagrams (30 Use Case, 30 Class, 30 Sequence)
Model evaluations	270 paired model–diagram observations
Models	OpenAI gpt-4.1; Anthropic claude-sonnet-5; Google gemini-3.1-pro-preview
Temperature	gpt-4.1: 0.2; claude-sonnet-5: provider default; gemini-3.1-pro-preview: 0.2
Top-p	gpt-4.1: 1.0; claude-sonnet-5: provider default; gemini-3.1-pro-preview: 1.0
Top-k	Provider default
Number of runs	One execution per diagram
Prompt strategy	One diagram-specific prompt per UML diagram type; identical across providers
Output format	JSON constrained by diagram-specific canonical schemas
Recorded artifacts	Raw response, parsed JSON, prompt and schema snapshots, hashes, timing, token counts, execution status, provider metadata, and manifests
Analysis	Python; deterministic normalization; paired nonparametric tests; $\alpha = .05$

Table 1. Experimental configuration and reproducibility settings used for the LLM-UML Benchmark v1.0 evaluation.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$F_1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (3)$$

3.3 Evaluation Framework and Statistical Analysis

We aggregated diagram-level metrics into the UML Understanding Assessment Framework (UUAUF). Every dimension ranges from 0 to 1, with higher values indicating closer agreement with the canonical gold standard. Structural Understanding is the mean F1 score for the primary elements of a diagram: actors and use cases; classes, attributes, and operations; or participants and messages. Relationship Understanding is the mean F1 score for applicable semantic links, including associations, include/extend/generalization links, typed class relationships, multiplicities, and message links. Behavioral Understanding applies only to Sequence diagrams and is the mean F1 score for message order and combined fragments. Functional Understanding is the F1 score for diagram-supported functional requirements after deterministic normalization. Representation Quality is the arithmetic mean of JSON validity, schema validity, and required-field completeness. Figure 5 illustrates the framework, and Appendix B supplies the diagram-specific formulas and aggregation rules.

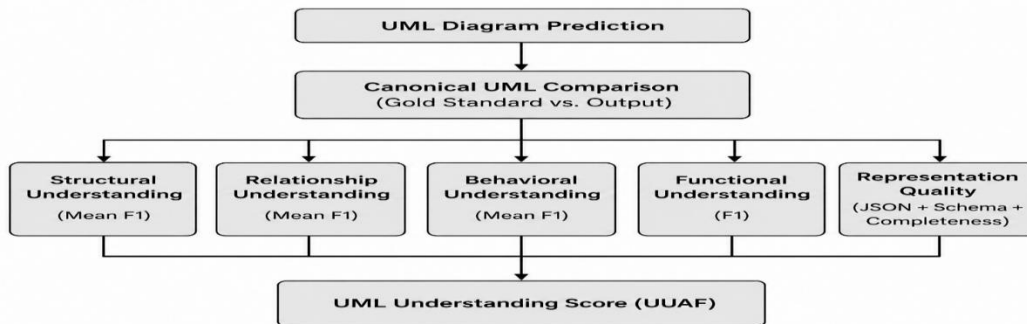


Figure 5. UML Understanding Assessment Framework (UUAUF). Diagram-level metrics are aggregated into five evaluation dimensions to compute the Composite UUAUF Score.

We calculated Representation Quality as the arithmetic mean of JSON validity (J), schema validity (S), and required-field completeness (C), where J and S are binary indicators and C represents the proportion of required schema fields present in the generated output.

$$RQ = \frac{J + S + C}{3} \quad (4)$$

We calculated each case's UML Understanding Score as the arithmetic mean of the applicable UUAUF dimensions.

$$UMLS_{core} = \frac{\sum \text{Applicable UUAUF Dimensions}}{k} \quad (5)$$

where k denotes the number of applicable UUAUF dimensions for the corresponding UML diagram type. Because Behavioral Understanding applies only to Sequence diagrams, we excluded that dimension from the denominator for Use Case and Class diagrams rather than assigning an artificial score of zero. Consequently, the composite score summarizes only the dimensions applicable to each UML diagram type.

We used paired nonparametric tests because all three models evaluated the same cases. Scores are bounded because each metric is restricted to the interval $[0, 1]$, rather than extending without limit. Tied observations occur when two or more models receive exactly the same score on a case, which was

common for near-perfect structural and representation scores. These features weaken the normality and continuous-measure assumptions of parametric repeated-measures tests. We therefore used the Friedman test (Friedman, 1937) for comparisons among the three models, Wilcoxon signed-rank tests (Wilcoxon, 1945) for pairwise comparisons, the Holm correction (Holm, 1979) to control the family-wise error rate, and Kendall's W (Kendall & Smith, 1939) as the omnibus effect size. We used $\alpha = .05$ and report descriptive and inferential statistics.

The diagram is the scoring unit, whereas the scenario is the clustering and complexity-assignment unit. The benchmark contains 30 scenarios—10 low-, 10 medium-, and 10 high-complexity scenarios—and each scenario is represented by one use case, one class, and one sequence diagram. Each diagram-specific analysis therefore includes 30 diagrams, with one diagram from each scenario and one matched set of scores from the three models. Thus, no scenario contributes repeated views within a diagram-specific test. In contrast, the pooled 90-diagram analysis includes three related notation-specific views of each scenario. Because these views share the same underlying scenario and complexity classification, they may exhibit within-scenario correlation and should not be treated as fully independent observations. We therefore treat the pooled omnibus result as a secondary summary, emphasize the diagram-specific paired analyses, and make no inferential claim of overall model superiority. The nonsignificant pooled result provides no statistical basis for an overall model ranking. Future work will evaluate robustness using scenario-level aggregation or cluster-resampling methods.

4. RESULTS

We evaluated three multimodal large language models across all 90 benchmark cases and analyzed their performance using the UML Understanding Assessment Framework (UAAF). Table 2 presents the model-level UAAF scores, and Table 3 summarizes the principal paired omnibus tests for the overall score, individual UAAF dimensions, and diagram-specific comparisons. Figures 6–9 present the principal benchmark results. Appendix C provides supplementary analyses of case-level score distributions, latency, token usage, benchmark wins, complexity patterns, and representation-quality diagnostics.

Model	Structural	Relationship	Behavioral*	Functional	Representation	Overall
Gemini 3.1 Pro Preview	.997	.964	.857	.232	.985	.801
Claude Sonnet 5	.996	.921	.989	.157	1.000	.780
GPT-4.1	.849	.876	.970	.272	1.000	.764

Table 2. Model-level UAAF scores. *Behavioral understanding applies to Sequence diagrams only.

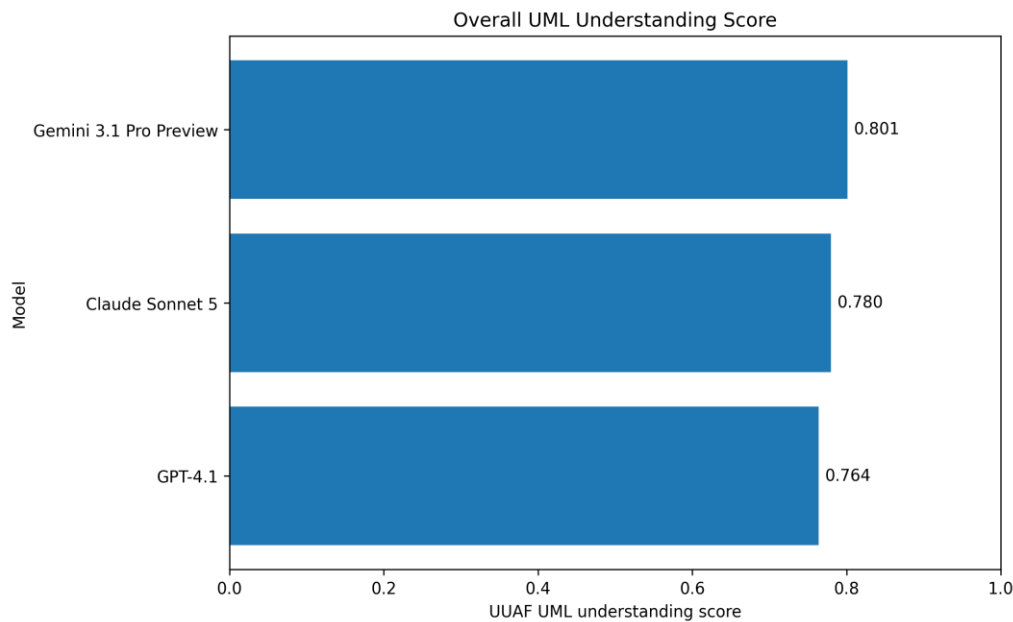


Figure 6. Descriptive overall Composite UAAF Scores across the complete 90-case benchmark.

4.1 RQ1: Structural Understanding

Gemini 3.1 Pro Preview achieved the highest Structural Understanding score (0.997), followed closely by Claude Sonnet 5 (0.996), whereas GPT-4.1 achieved 0.849. Figure 7 presents the complete UUAF dimension profile. The Friedman test indicated statistically significant differences among the models, $\chi^2(2) = 45.371$, $p < .001$, Kendall's $W = .252$. Holm-adjusted pairwise comparisons showed that both Gemini and Claude significantly outperformed GPT-4.1, whereas the difference between Gemini and Claude was not statistically significant.

4.2 RQ2: Relationship Understanding

Gemini achieved the highest Relationship Understanding score (0.964), followed by Claude Sonnet 5 (0.921) and GPT-4.1 (0.876). The Friedman test identified statistically significant differences among the models, $\chi^2(2) = 54.931$, $p < .001$, Kendall's $W = .305$. All Holm-adjusted pairwise comparisons remained statistically significant, indicating consistent differences in relationship interpretation across the three models.

4.3 RQ3: Behavioral Understanding

Behavioral Understanding was evaluated only for the 30 Sequence diagrams because behavioral semantics are defined exclusively for UML Sequence diagrams. Claude Sonnet 5 achieved the highest score (0.989), followed by GPT-4.1 (0.970) and Gemini 3.1 Pro Preview (0.857). A Friedman test conducted on the 30 paired Sequence cases identified statistically significant differences among the models, $\chi^2(2)=10.400$, $p=.006$, Kendall's $W=.173$. Holm-adjusted pairwise comparisons showed that Claude Sonnet 5 significantly outperformed Gemini 3.1 Pro Preview, whereas the difference between Claude Sonnet 5 and GPT-4.1 was not statistically significant. GPT-4.1 also significantly outperformed Gemini 3.1 Pro Preview. Overall, all three models demonstrated strong behavioral understanding, although Claude Sonnet 5 achieved the strongest performance in preserving message ordering and interaction semantics.

4.4 RQ4: Functional Understanding

Functional Understanding was the weakest UUAF dimension across all three models. Although GPT-4.1 significantly outperformed the other models, Functional Understanding remained the weakest UUAF dimension across all three models. The Friedman test identified statistically significant differences, $\chi^2(2) = 7.849$, $p = .020$, Kendall's $W = .044$. Holm-adjusted comparisons showed that GPT-4.1 significantly outperformed both Claude and Gemini, whereas the difference between Claude and Gemini was not statistically significant. These findings indicate that the models extracted explicit UML content more reliably than they generated functional requirements that aligned with the creator-developed canonical reference under deterministic matching. Appendix C presents this pattern separately by diagram type.

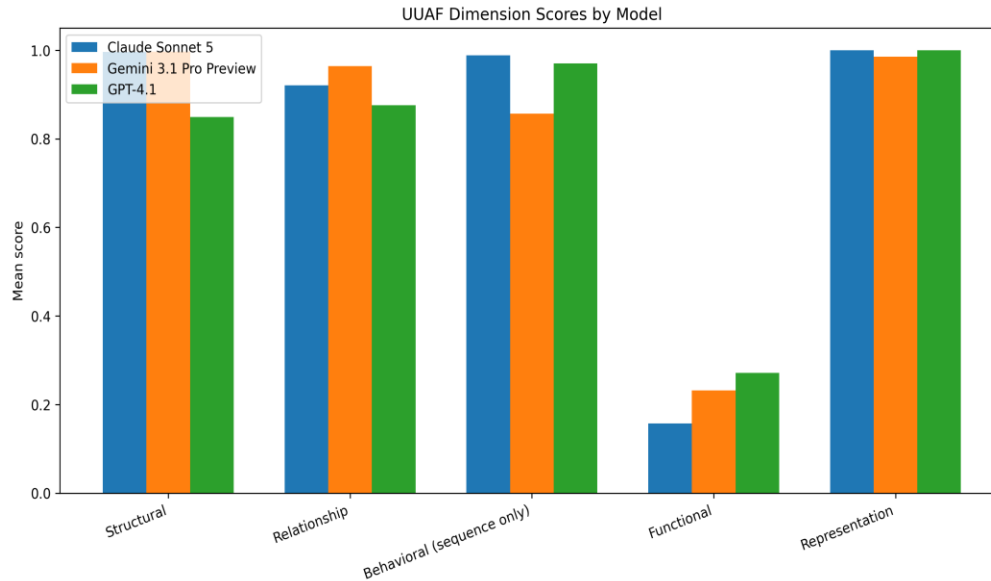
4.5 RQ5: Machine-Processable Structured Representations

All three models generated valid JSON for all 90 benchmark cases. Claude Sonnet 5 and GPT-4.1 produced schema-compliant outputs for every case. Gemini generated schema-compliant outputs for 86 cases and produced four schema-validation errors. Consequently, Claude and GPT-4.1 each achieved a Representation Quality score of 1.000, whereas Gemini achieved 0.985. These findings indicate that all three models generally produced machine-processable outputs suitable for automated evaluation, with only minor schema-conformance errors observed for Gemini. Because Representation Quality exhibited near-ceiling performance and minimal variability, the results are interpreted descriptively.

4.6 RQ6: Variation Across Models, UML Diagram Types, and Benchmark Complexity

Gemini achieved the highest overall UML Understanding Score ($M = 0.801$), followed by Claude Sonnet 5 ($M = 0.780$) and GPT-4.1 ($M = 0.764$), as shown in Figure 6. However, the overall differences were not statistically significant, $\chi^2(2) = 4.738$, $p = .094$, Kendall's $W = .026$. Holm-adjusted pairwise comparisons were also nonsignificant. Accordingly, the overall model ordering should be interpreted descriptively rather than as evidence of statistically reliable superiority.

Performance varied substantially across UML diagram types. As shown in Figure 8, GPT-4.1 achieved the highest mean score for Use Case diagrams, Gemini achieved the highest mean score for Class diagrams, and Claude achieved the highest mean score for Sequence diagrams. Friedman tests identified statistically significant model differences for Use Case diagrams, $\chi^2(2) = 40.653$, $p < .001$, Kendall's $W = .678$; Class diagrams, $\chi^2(2) = 32.000$, $p < .001$, Kendall's $W = .533$; and Sequence diagrams, $\chi^2(2) = 18.121$, $p < .001$, Kendall's $W = .302$. These results demonstrate that aggregate scores conceal meaningful model specialization across UML diagram types.



**Figure 7. Mean UAAF dimension scores by model;
Behavioral Understanding applies only to Sequence diagrams.**

Performance generally declined as benchmark complexity increased, although the pattern was not uniform across models. Figure 9 shows that Claude decreased from 0.812 on low-complexity cases to 0.770 on medium-complexity cases and 0.758 on high-complexity cases. Gemini decreased from 0.848 to 0.784 and 0.773, respectively, whereas GPT-4.1 decreased from 0.821 to 0.739 and 0.732. GPT-4.1 therefore exhibited the largest descriptive reduction between low- and high-complexity cases. Between-model differences within individual complexity levels were not statistically significant; therefore, the complexity findings should be interpreted descriptively and in conjunction with diagram type.

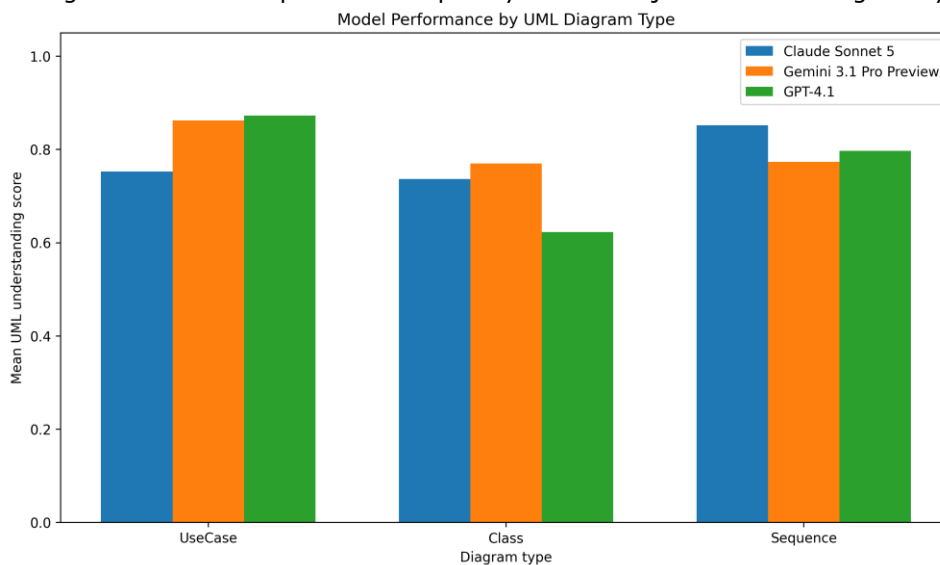


Figure 8. Mean Composite UAAF Score by UML diagram type and model.

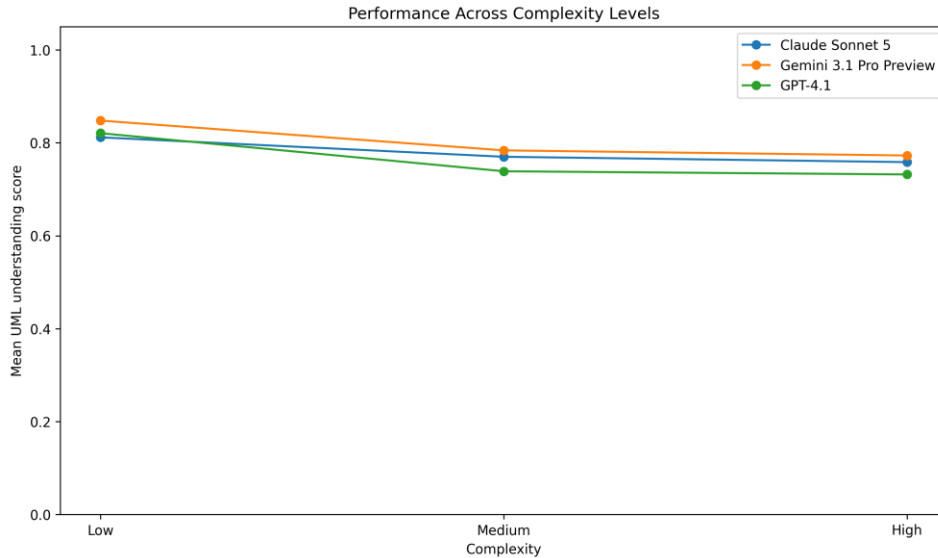


Figure 9. Mean Composite UAF Score across low, medium, and high benchmark complexity.

Comparison	Statistic	p	Effect/Decision
Overall score	Friedman $\chi^2(2)=4.738$	.094	W=.026; not significant
Structural	Friedman $\chi^2(2)=45.371$	<.001	W=.252; significant
Relationship	Friedman $\chi^2(2)=54.931$	<.001	W=.305; significant
Behavioral	Friedman $\chi^2(2)=10.400$	.006	W=.173; significant
Functional	Friedman $\chi^2(2)=7.849$	.020	W=.044; significant
Use Case overall	Friedman $\chi^2(2)=40.653$	<.001	W=.678; significant
Class overall	Friedman $\chi^2(2)=32.000$	<.001	W=.533; significant
Sequence overall	Friedman $\chi^2(2)=18.121$	<.001	W=.302; significant

Table 3. Principal paired omnibus statistical tests.

5. DISCUSSION

The results show that LLM-UML Benchmark distinguishes model capabilities that an overall leaderboard cannot capture. Although Gemini achieved the highest descriptive composite score, the nonsignificant omnibus comparison prevents us from claiming overall superiority. Instead, the benchmark revealed meaningful differences across UML Understanding dimensions and diagram types. Current multimodal models demonstrate distinct strengths in interpreting Use Case, Class, and Sequence diagrams, supporting conceptual-modeling research that treats UML notations as cognitively and semantically distinct rather than interchangeable (Gemino & Wand, 2005; Moody, 2009; Purchase et al., 2001).

Gemini and Claude achieved near-perfect Structural Understanding, indicating that three multimodal models can reliably identify explicit UML elements. However, lower Relationship and Behavioral Understanding scores show that correctly identifying diagram elements does not necessarily imply complete semantic interpretation. Correctly interpreting multiplicities, relationship types, message ordering, and combined fragments requires notation-aware reasoning beyond visual recognition, consistent with prior studies on UML comprehension (Lange & Chaudron, 2006; Lange et al., 2006).

Functional Understanding emerged as the most challenging evaluation dimension. All three models scored substantially lower on Functional Understanding than on Structural Understanding, indicating that inferring intended system functionality remains more difficult than extracting explicit diagram content. Two representative scoring cases illustrate the problem. First, if a Sequence diagram explicitly shows appointment creation and confirmation but a model returns only “the system manages appointments,” the generic statement omits two traceable behaviors and lowers recall. Second, if a Class diagram contains Customer and Order structures but the model adds “the system sends delivery notifications” without a supporting operation or interaction, the unsupported requirement is counted as a false positive and lowers precision. An AI assistant may therefore identify actors, classes, or messages

correctly while still producing incomplete, overly general, or unsupported requirements. We recommend multidimensional reporting rather than reliance on one overall score.

Our findings also demonstrate the value of standardized benchmark design. Canonical prompts, diagram-specific schemas, deterministic normalization, and creator-developed canonical reference enabled reproducible comparisons across models, while preserved outputs and execution artifacts supported transparency and auditability. The four Gemini schema-validation errors further demonstrate that Representation Quality should remain an explicit evaluation dimension because schema violations increase downstream integration effort, even when much of the semantic content is correct.

Beyond the empirical evaluation, LLM-UML Benchmark provides a reusable research and educational resource. The benchmark contributes 30 Systems Analysis and Design scenarios created by one SME and reviewed and evaluated by two additional SMEs, instantiated as 90 UML diagrams across multiple domains and complexity levels. By releasing aligned PlantUML source files, UML images, canonical schemas, gold standards, prompts, metadata, and evaluation tools, we provide a common foundation for reproducible research, standardized model comparison, and AI-supported Systems Analysis and Design education.

6. LIMITATIONS

We acknowledge several limitations. First, the benchmark includes only use case, class, and sequence diagrams, so the findings may not generalize to other UML notations. Second, the scenarios are controlled educational cases rather than industrial models, which supports experimental consistency but limits ecological validity. Third, each model was executed only once per diagram; therefore, we did not estimate within-model variability. Fourth, we evaluated three commercial models and excluded open-weight and locally deployed models. Fifth, the pooled 90-diagram analysis includes three views of each scenario and may retain within-scenario dependence. Diagram-specific tests avoid repeated views, so we base our substantive conclusions primarily on those analyses. Sixth, SME ratings indicate that visual quality declined as assigned complexity increased and that all revision recommendations concerned high-complexity diagrams. Consequently, observed complexity effects may reflect both semantic complexity and differences in visual presentation quality. Finally, the study evaluates only Task 1 (Diagram-to-Structured-UML).

The canonical references were authored by the benchmark creator and were not independently reconstructed element by element by the two reviewing SMEs. Although the reviewers evaluated diagram correctness, coherence, and answer-key interpretability, this process does not establish expert-consensus validation of every canonical UML element or functional requirement. Functional Understanding was also measured by comparing model-generated requirements with these creator-developed canonical references using deterministic normalization and matching. Consequently, semantically valid alternative requirements may receive limited credit, and Functional Understanding should be interpreted as alignment with the canonical references under the specified evaluation procedure rather than as a complete measure of requirements-understanding ability. Future benchmark releases will incorporate independent reference reconstruction, discrepancy adjudication, element-level expert-agreement reporting, reviewed alternative phrasings, requirement-equivalence classes, and a documented semantic-matching sensitivity analysis.

Because Behavioral Understanding applies only to sequence diagrams, Composite UUAF Scores should be interpreted alongside their component dimensions rather than as fully uniform measures across all diagram types. In addition, latency and token usage depend on provider infrastructure, model implementation, and provider-specific tokenization; therefore, they should not be interpreted as direct measures of computational efficiency or monetary cost.

7. FUTURE WORK

We will extend LLM-UML Benchmark in four directions. First, we will validate the remaining tasks, including Diagram-to-Functional Requirements, Diagram-to-PlantUML, Diagram-to-Natural Language Explanation, Diagram Question Answering, and Cross-Diagram Reasoning. Second, we will add further UML notations, industrial and student-created diagrams, open-weight multimodal models, repeated executions, human baselines, and scenario-level sensitivity analyses. Third, we will refine Functional

Understanding through reviewed alternative gold-standard requirements, explicit traceability links, expert agreement reporting, and comparisons of deterministic and semantic matching. Finally, we will establish a versioned leaderboard and examine AI-supported UML interpretation in controlled classroom studies.

8. CONCLUSION

We introduced LLM-UML Benchmark v1.0, a specification-centered and reproducible benchmark for evaluating multimodal large language models on UML diagram understanding. The benchmark includes 30 scenarios created by one SME and reviewed and evaluated by two additional SMEs, instantiated as 90 balanced UML diagrams, together with aligned PlantUML source files, UML images, canonical schemas, gold standards, prompts, metadata, and evaluation tools. Our empirical evaluation showed that Gemini achieved the highest descriptive overall Composite UUAF Score, although the overall model differences were not statistically significant. More importantly, the benchmark revealed distinct strengths across UML diagram types and evaluation dimensions, while all three models found Functional Understanding substantially more challenging than explicit structural interpretation. LLM-UML Benchmark provides a foundation for standardized model comparison, reproducible research, and AI-supported Systems Analysis and Design education, while supporting future benchmark tasks and additional UML diagram types.

9. ACKNOWLEDGEMENTS

The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by grant funding from the National Science Foundation (NSF) for the collaborative project titled "CAP: Expanding Use-Inspired AI Research and Instruction Across Campus," Award #2401658. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

The authors also acknowledge the developers and maintainers of the open source tools and software libraries, including PlantUML, that supported UML diagram generation, processing, and evaluation. The benchmark materials are provided to promote transparency, reproducibility, and future research.

10. REFERENCES

- Agarwal, R., & Sinha, A. P. (2003). Object-oriented modeling with UML: A study of developers' perceptions. *Communications of the ACM*, 46(9), 248-256. <https://doi.org/10.1145/903893.903944>
- Agarwal, R., De, P., & Sinha, A. P. (1999). Comprehending object and process models: An empirical study. *IEEE Transactions on Software Engineering*, 25(4), 541-556. <https://doi.org/10.1109/32.799953>
- Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4), 105-120. <https://doi.org/10.1609/aimag.v35i4.2513>
- Arpteg, A., Brinne, B., Crnkovic-Friis, L., & Bosch, J. (2018). Software engineering challenges of deep learning. 2018 44th Euromicro Conference on Software Engineering and Advanced Applications, 50-59. <https://doi.org/10.1109/SEAA.2018.00018>
- Bavota, G., & Russo, B. (2016). A large-scale empirical study on self-admitted technical debt. 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories, 315-326. <https://doi.org/10.1145/2901739.2901742>
- Bates, A., Vavricka, R., Carleton, S., Shao, R., & Pan, C. (2025). Unified modeling language code generation from diagram images using multimodal large language models. *Machine Learning with Applications*, 20, 100660. <https://doi.org/10.1016/j.mlwa.2025.100660>

- Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023). Programming is hard—or at least it used to be: Educational opportunities and challenges of AI code generation. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 500-506. <https://doi.org/10.1145/3545945.3569759>
- Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61-72. <https://doi.org/10.1109/2.59>
- Briand, L. C., Labiche, Y., & O'Sullivan, L. (2003). Impact analysis and change management of UML models. *International Conference on Software Maintenance*, 256-265. <https://doi.org/10.1109/ICSM.2003.1235428>
- Budgen, D., Burn, A. J., Brereton, O. P., Kitchenham, B. A., & Pretorius, R. (2011). Empirical evidence about the UML: A systematic literature review. *Software: Practice and Experience*, 41(4), 363-392. <https://doi.org/10.1002/spe.1009>
- Buse, R. P. L., & Weimer, W. R. (2010). Learning a metric for code readability. *IEEE Transactions on Software Engineering*, 36(4), 546-558. <https://doi.org/10.1109/TSE.2009.70>
- Cabot, J., & Gogolla, M. (2012). Object constraint language (OCL): A definitive guide. In *Formal Methods for Model-Driven Engineering* (pp. 58-90). Springer. https://doi.org/10.1007/978-3-642-30982-3_3
- Cámara, J., Burgueño, L., & Troya, J. (2024). Towards standardized benchmarks of LLMs in software modeling tasks: A conceptual framework. *Software and Systems Modeling*, 23, 1309–1318. <https://doi.org/10.1007/s10270-024-01206-9>
- Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., ... Zaremba, W. (2021). Evaluating large language models trained on code. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, 1-35.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., & Tafjord, O. (2018). Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *Proceedings of the 2018 Workshop on Machine Reading for Question Answering*, 1-11.
- Daun, M., Brings, J., Obe, P. A., & Weyer, T. (2017). Teaching conceptual modeling in online courses: Coping with the need for individual feedback to modeling exercises. *2017 IEEE 30th Conference on Software Engineering Education and Training*, 134-143
- Dennis, A., Wixom, B. H., & Roth, R. M. (2020). *Systems analysis and design* (7th ed.). Wiley.
- Denny, P., Kumar, V., & Giacaman, N. (2023). Conversing with Copilot: Exploring prompt engineering for solving CS1 problems using natural language. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 1136-1142. <https://doi.org/10.1145/3545945.3569823>
- Dobing, B., & Parsons, J. (2006). How UML is used. *Communications of the ACM*, 49(5), 109-113. <https://doi.org/10.1145/1125944.1125949>
- Doris, A. C., Grandi, D., Tomich, R., Alam, M. F., Ataei, M., Cheong, H., & Ahmed, F. (2025). DesignQA: A multimodal benchmark for evaluating large language models' understanding of engineering documentation. *Journal of Computing and Information Science in Engineering*, 25(2), 021009. <https://doi.org/10.1115/1.4067333>
- Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of OpenAI Codex on introductory programming. *Proceedings of the 24th Australasian Computing Education Conference*, 10-19. <https://doi.org/10.1145/3511861.3511863>

- France, R., & Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. *Future of Software Engineering*, 37-54. <https://doi.org/10.1109/FOSE.2007.14>
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200), 675-701. <https://doi.org/10.1080/01621459.1937.10503522>
- Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering*, 9, 248-260. <https://doi.org/10.1007/s00766-004-0204-6>
- Gemino, A., & Wand, Y. (2005). Complexity and clarity in conceptual modeling: Comparison of mandatory and optional properties. *Data & Knowledge Engineering*, 55(3), 301-326. <https://doi.org/10.1016/j.datak.2004.12.009>
- Genero, M., Piattini, M., & Calero, C. (2005). A survey of metrics for UML class diagrams. *Journal of Object Technology*, 4(9), 59-92.
- GitHub. (2026a). Software-requirement-specification [GitHub topic]. Retrieved June 1, 2026, from <https://github.com/topics/software-requirement-specification>
- GitHub. (2026b). Srs-document [GitHub topic]. Retrieved June 1, 2026, from <https://github.com/topics/srs-document>
- Glinz, M. (2000). Problems and deficiencies of UML as a requirements specification language. *Proceedings of the 10th International Workshop on Software Specification and Design*, 11-22.
- Hassine, J. (2025). Evaluating multi-modal LLMs for automatically recognizing semantic elements in UML use case diagram images. *Proceedings of the 2025 IEEE International Conference on Software Analysis, Evolution and Reengineering*, 861-866. <https://doi.org/10.1109/SANER64311.2025.00092>
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2021). Measuring massive multitask language understanding. *International Conference on Learning Representations*.
- Hindle, A., Barr, E. T., Su, Z., Gabel, M., & Devanbu, P. (2012). On the naturalness of software. 2012 34th International Conference on Software Engineering, 837-847. <https://doi.org/10.1109/ICSE.2012.6227135>
- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2), 65-70.
- Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2024). Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8), Article 220. <https://doi.org/10.1145/3695988>
- Hutchinson, J., Whittle, J., Rouncefield, M., & Kristoffersen, S. (2011). Empirical assessment of MDE in industry. 2011 33rd International Conference on Software Engineering, 471-480. <https://doi.org/10.1145/1985793.1985858>
- Ibáñez, M. B., Barrón-Estrada, M. L., & Zatarain-Cabada, R. (2025). Can multimodal large language models grade like an expert? A study on UML class diagram assessment accuracy. *Computer Applications in Engineering Education*, 33(5), e70080. <https://doi.org/10.1002/cae.70080>
- Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). SWE-bench: Can language models resolve real-world GitHub issues? *International Conference on Learning Representations*.
- Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming.

- Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, Article 42.
<https://doi.org/10.1145/3544548.3580919>
- Kendall, M. G., & Smith, B. B. (1939). The problem of m rankings. *The Annals of Mathematical Statistics*, 10(3), 275-287. <https://doi.org/10.1214/aoms/1177732186>
- Krogstie, J., Sindre, G., & Jorgensen, H. (2006). Process models representing knowledge for action: A revised quality framework. *European Journal of Information Systems*, 15(1), 91-102. <https://doi.org/10.1057/palgrave.ejis.3000598>
- Lange, C. F. J., & Chaudron, M. R. V. (2006). Effects of defects in UML models: An experimental investigation. 28th International Conference on Software Engineering, 401-411. <https://doi.org/10.1145/1134285.1134341>
- Lange, C. F. J., DuBois, B., Chaudron, M. R. V., & Demeyer, S. (2006). An experimental investigation of UML modeling conventions. *Model Driven Engineering Languages and Systems*, 27-41. https://doi.org/10.1007/11880240_3
- Lindland, O. I., Sindre, G., & Solvberg, A. (1994). Understanding quality in conceptual modeling. *IEEE Software*, 11(2), 42-49. <https://doi.org/10.1109/52.268955>
- Lu, S., Guo, D., Ren, S., Huang, J., Svyatkovskiy, A., Blanco, A., Clement, C., Drain, D., Jiang, D., Tang, D., Li, G., Zhou, L., Shou, L., Zhou, L., Tufano, M., Gong, M., Zhou, M., Duan, N., & Sundaresan, N. (2021). CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, 1-21.
- Ma, Z., Zhang, B., Zhang, J., Yu, J., Zhang, X., Li, Z., & Zhang, S. (2024). SpreadsheetBench: Towards challenging real-world spreadsheet manipulation. *Advances in Neural Information Processing Systems*, 37.
- Mellor, S. J., Clark, A. N., & Futagami, T. (2003). Model-driven development. *IEEE Software*, 20(5), 14-18. <https://doi.org/10.1109/MS.2003.1231145>
- Meyer, M., Wiederkehr, I., Panzner, M., Koldewey, C., & Dumitrescu, R. (2022). A Reference Process Model for Usage Data-Driven Product Planning.
- Mohagheghi, P., Dehlen, V., & Neple, T. (2009). Definitions and approaches to model quality in model-based software development: A review of literature. *Information and Software Technology*, 51(12), 1646-1669. <https://doi.org/10.1016/j.infsof.2009.04.004>
- Moody, D. L. (2009). The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. *IEEE Transactions on Software Engineering*, 35(6), 756-779. <https://doi.org/10.1109/TSE.2009.67>
- Nugroho, A., & Chaudron, M. R. V. (2008). A survey into the rigor of UML use and its perceived impact on quality and productivity. *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 90-99. <https://doi.org/10.1145/1414004.1414020>
- Paas, F., Renkl, A., & Sweller, J. (2003). Cognitive load theory and instructional design: Recent developments. *Educational Psychologist*, 38(1), 1-4. https://doi.org/10.1207/S15326985EP3801_1
- Petre, M. (2013). UML in practice. 2013 35th International Conference on Software Engineering, 722-731. <https://doi.org/10.1109/ICSE.2013.6606618>
- Purchase, H. C., Colpoys, L., McGill, M., Carrington, D., & Britton, C. (2001). UML class diagram syntax: An empirical study of comprehension. *Proceedings of the 2001 Asia-Pacific Symposium on Information Visualisation*, 113-120.

- Recker, J., Rosemann, M., Indulska, M., & Green, P. (2009). Business process modeling-a comparative analysis. *Journal of the Association for Information Systems*, 10(4), 333-363. <https://doi.org/10.17705/1jais.00193>
- Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2015). *Systems analysis and design in a changing world* (6th ed.). Cengage Learning.
- Sedrakyan, G., & Snoeck, M. (2016). Cognitive feedback and behavioral feedforward automation perspectives for modeling and validation in a learning context. *Information and Software Technology*, 83, 91-102.
- Siau, K., & Loo, P. P. (2006). Identifying difficulties in learning UML. *Information Systems Management*, 23(3), 43-51. <https://doi.org/10.1201/1078.10580530/46108.23.3.20060601/93706.5>
- Siau, K., & Rossi, M. (2011). Evaluation techniques for systems analysis and design modelling methods: A review and comparative analysis. *Information Systems Journal*, 21(3), 249-268. <https://doi.org/10.1111/j.1365-2575.2007.00255.x>
- Storrie, H. (2017). How are conceptual models used in industrial software development? A descriptive survey. 2017 21st International Conference on Engineering of Complex Computer Systems, 160-169.
- Sommerville, I. (2016). *Software engineering* (10th ed.). Pearson.
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2), 257-285. https://doi.org/10.1207/s15516709cog1202_4
- Tufano, M., Watson, C., Bavota, G., Di Penta, M., White, M., & Poshyanyk, D. (2019). An empirical study on learning bug-fixing patches in the wild via neural machine translation. *ACM Transactions on Software Engineering and Methodology*, 28(4), Article 19. <https://doi.org/10.1145/3340544>
- Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022). Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, Article 332. <https://doi.org/10.1145/3491101.3519665>
- Wand, Y., & Weber, R. (2002). Research commentary: Information systems and conceptual modeling-a research agenda. *Information Systems Research*, 13(4), 363-376. <https://doi.org/10.1287/isre.13.4.363.69>
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6), 80-83. <https://doi.org/10.2307/3001968>
- Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education*, 26(4), 235-254. <https://doi.org/10.1080/08993408.2016.1257418>
- Zhuo, T. Y., Huang, Y., Chen, C., & Xing, Z. (2023). Red teaming chatgpt via jailbreaking: Bias, robustness, reliability and toxicity. *arXiv preprint arXiv:2301.12867*.

APPENDIX A

Reproducibility Workflow

The complete benchmark artifact is publicly available at "<https://anonymous.4open.science/r/LLM-UML-Benchmark-F577/README.md>". The repository includes the benchmark specification, canonical prompts and schemas, benchmark metadata, gold standards, provider adapters, evaluation framework, statistical analysis pipeline, documentation, and environment requirements. Benchmark cases and experiment outputs are organized according to the repository specification to support transparent and reproducible evaluation.

A complete reproduction consists of six steps:

1. Install the pinned Python environment and project dependencies.
2. Configure provider API keys using environment variables without committing secrets.
3. Validate the benchmark metadata and gold standards.
4. Execute Task 1 for each supported provider using the canonical prompts and schemas.
5. Evaluate the generated outputs using the deterministic evaluator and the UML Understanding Assessment Framework (UUAF).
6. Run the benchmark analysis pipeline to reproduce the statistical analyses, benchmark leaderboard, and publication-quality figures reported in this paper.

The repository records benchmark version information, model identifiers, provider names, execution timestamps, and evaluation metadata so that future benchmark runs can distinguish benchmark revisions from changes in underlying model implementations. All benchmark documentation, evaluation protocols, and statistical procedures are included in the public repository to support independent verification and future benchmark extensions.

APPENDIX B

Operational Definitions and Aggregation of the UML Understanding Assessment Framework (UUAF)

This appendix describes the operational definitions and aggregation procedures used to compute the five dimensions of the UML Understanding Assessment Framework (UUAF). We calculated every metric deterministically from normalized benchmark outputs without using semantic embeddings, fuzzy matching, synonym dictionaries, or another large language model as an evaluator. This design ensures that every reported score is transparent, reproducible, and directly traceable to the benchmark outputs.

B.1 Structural Understanding

Structural Understanding measures the extent to which a model correctly identifies the primary modeling elements explicitly represented in a UML diagram. Because each UML diagram type contains different structural concepts, we calculated Structural Understanding separately for each diagram type. For Use Case diagrams, Structural Understanding equals the mean F1 score for actors and use cases.

$$SU_{UC} = \frac{F1_{Actors} + F1_{UseCases}}{2}$$

For Class diagrams, Structural Understanding equals the mean F1 score for classes, attributes, and operations.

$$SU_{Class} = \frac{F1_{Classes} + F1_{Attributes} + F1_{Operations}}{3}$$

For Sequence diagrams, Structural Understanding equals the mean F1 score for participants and messages.

$$SU_{Seq} = \frac{F1_{Participants} + F1_{Messages}}{2}$$

B.2 Relationship Understanding

Relationship Understanding measures how accurately a model preserves semantic relationships among UML elements.

For Use Case diagrams, we calculated Relationship Understanding as the mean F1 score for actor–use case associations, include relationships, extend relationships, and generalizations.

$$RU_{UC} = \frac{F1_{Association} + F1_{Include} + F1_{Extend} + F1_{Generalization}}{4}$$

For Class diagrams, we calculated Relationship Understanding as the mean F1 score for exact relationships, relationship types, and multiplicities.

$$RU_{Class} = \frac{F1_{ExactRelationship} + F1_{RelationshipType} + F1_{Multiplicity}}{3}$$

For Sequence diagrams, we calculated Relationship Understanding as the mean F1 score for message identification and message types.

$$RU_{Seq} = \frac{F1_{MessageLinks} + F1_{MessageType}}{2}$$

B.3 Behavioral Understanding

Behavioral Understanding applies only to Sequence diagrams because Use Case and Class diagrams do not directly represent execution behavior. We calculated Behavioral Understanding as the mean F1 score for message ordering and combined fragments.

$$BU = \frac{F1_{MessageOrder} + F1_{CombinedFragments}}{2}$$

For Use Case and Class diagrams, Behavioral Understanding was reported as not applicable rather than assigned a value of zero.

B.4 Functional Understanding

Functional Understanding measures how accurately a model identifies the intended system functionality represented by a UML diagram. We calculated Functional Understanding as the F1 score obtained by comparing normalized model-generated functional requirements with the creator-developed canonical reference.

$$FU = F1_{Requirements}$$

B.5 Representation Quality

Representation Quality measures whether the generated output is valid, complete, and suitable for automated evaluation. We calculated Representation Quality as the arithmetic mean of JSON validity (J), schema validity (S), and required-field completeness (C).

$$RQ = \frac{J + S + C}{3}$$

where:

- $J = 1$ if the generated output is valid JSON; otherwise $J = 0$;
- $S = 1$ if the generated output conforms to the canonical schema; otherwise $S = 0$;
- C is the proportion of required schema fields present in the generated output, where $0 \leq C \leq 1$.

B.6 Composite UAAF Score

We calculated the Composite UAAF Score as the arithmetic mean of the applicable UAAF dimensions. For Sequence diagrams, the score includes all five dimensions.

$$UMLScore_{seq} = \frac{SU + RU + BU + FU + RQ}{5}$$

For Use Case and Class diagrams, Behavioral Understanding is not applicable; therefore, we excluded it from the denominator.

$$UMLScore_{uc/class} = \frac{SU + RU + FU + RQ}{4}$$

We report the five UAAF dimensions as the primary evaluation outcomes throughout the paper. We use the Composite UAAF Score only as a summary measure to facilitate overall comparison across models and UML diagram types. We assign equal weight to each applicable dimension, and we do not substitute missing dimensions with zero values. This aggregation strategy reflects the semantic characteristics of each UML diagram type while preserving comparability across benchmark cases.

APPENDIX C

Additional Experimental Results

This appendix presents supplementary analyses that support the findings in Section 4. We include case-level distributions, inference latency, token usage, functional performance by diagram type, case-level wins, accuracy-latency trade-offs, diagram-complexity heatmaps, and required-field completeness.

C.1 Case-Level Score Distributions

We observed substantial overlap among the three model distributions despite different mean scores. The distributions show why the nonsignificant overall Friedman result is compatible with meaningful diagram-specific differences. Figure C1 illustrates the distribution of Composite UUAF Scores across the 90 benchmark cases for each evaluated model. The distributions demonstrate that performance variability differed across providers, reflecting the heterogeneous difficulty of the benchmark cases. Although the average benchmark scores were relatively close, the distributions reveal differences in consistency that are not apparent from the mean values alone.

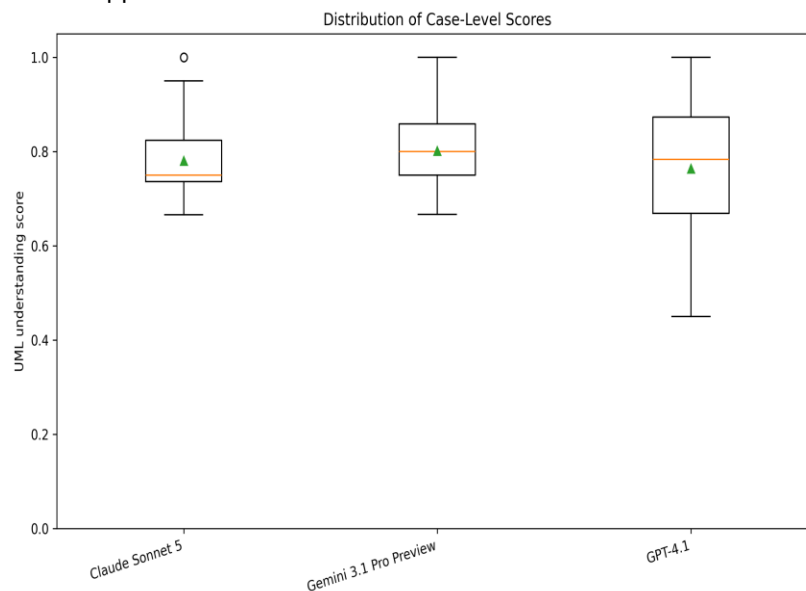


Figure C1. Distribution of case-level Composite UUAF Scores by model.

C.2 Average Inference Latency

We report latency as a practical reproducibility measure rather than a model-quality outcome. Provider infrastructure and network conditions contributed to the observed execution times. Figure C2 compares the average inference latency for the three evaluated models. This analysis complements the accuracy results by illustrating the computational cost associated with each provider. Together with the benchmark scores, these results highlight the practical trade-off between execution time and UML interpretation performance.

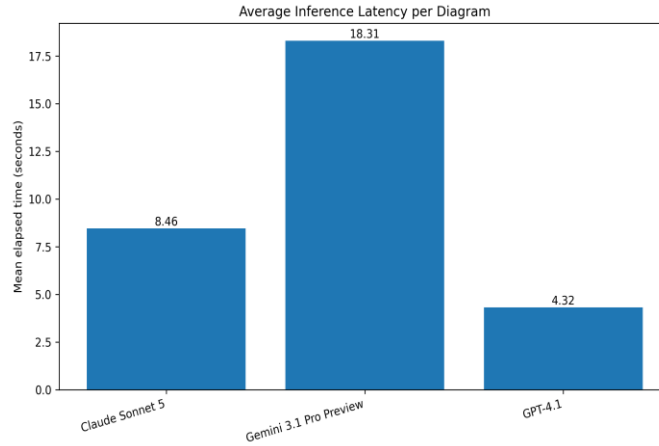


Figure C2. Average inference latency per UML diagram.

C.3 Average Token Usage

We report provider-recorded token usage to support cost and efficiency analysis. Token counts are not perfectly comparable because providers use different tokenization systems. Figure C3 summarizes the average input and output token usage across the benchmark. Reporting token consumption improves reproducibility and enables future researchers to compare both benchmark performance and computational cost when evaluating additional models.

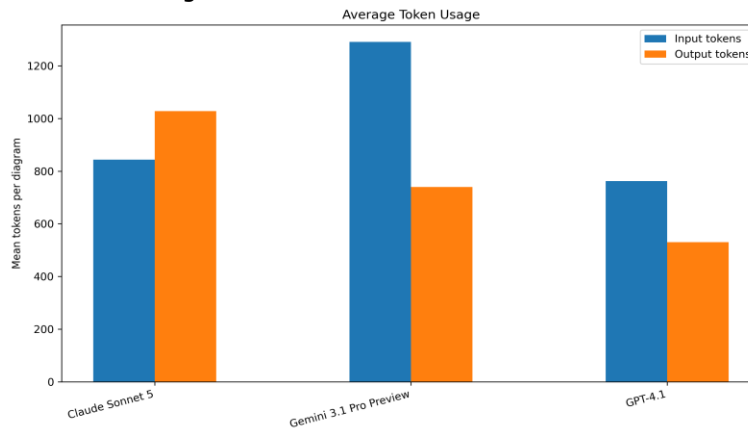


Figure C3. Average input and output token usage per UML diagram.

C.4 Functional Understanding by Diagram Type

Functional performance remained lower than structural and relationship performance across diagram types, reinforcing the conclusion that deterministic requirement alignment remains the benchmark's most difficult dimension. Figure C4 presents Functional Understanding separately for Use Case, Class, and Sequence diagrams. The results demonstrate that functional-requirement inference remained consistently more challenging than structural extraction across all diagram types, supporting the findings presented in Section 4.

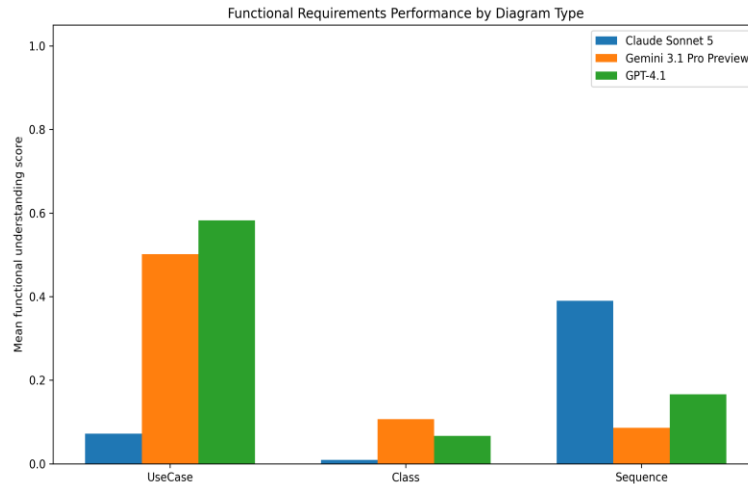


Figure C4. Mean Functional Understanding by UML diagram type and model.

C.5 Case-Level Benchmark Wins

Case-level wins show that no model dominated every scenario. This pattern further supports the use of diagram-specific and dimension-specific reporting. Figure C5 reports the number of benchmark cases for which each model achieved the highest Composite UAAF Score. This analysis complements the average benchmark results by illustrating how frequently each model performed best on individual benchmark cases.

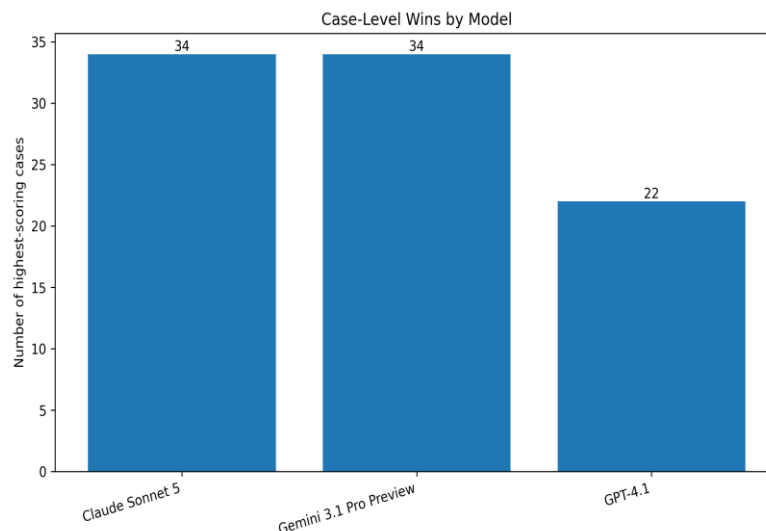


Figure C5. Number of benchmark cases won by each model.

C.6 Accuracy-Latency Trade-off

The scatter plot shows substantial within-model variation and illustrates that higher latency did not guarantee a higher case-level score. Figure C6 relates case-level Composite UAAF Scores to case-level inference latency. The figure illustrates the trade-off between benchmark performance and computational efficiency, enabling researchers to evaluate model quality alongside practical deployment considerations.

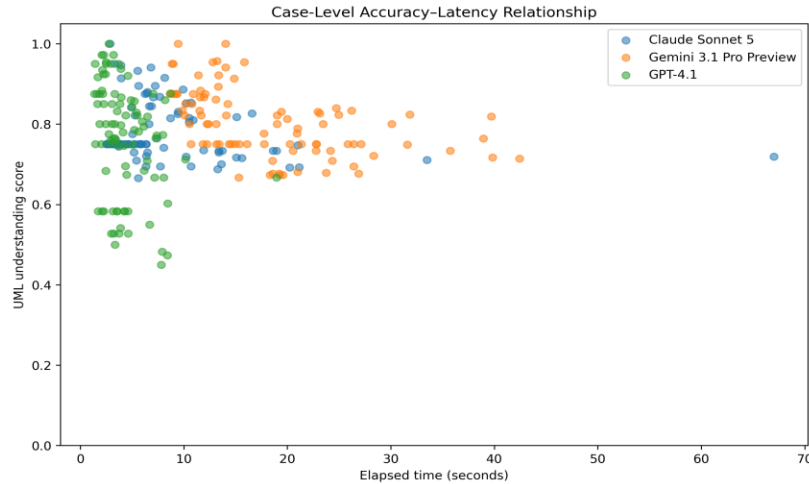


Figure C6. Relationship between case-level Composite UUAF Score and inference latency.

C.7 Diagram-Complexity Heatmaps

Figures C7a–C7c visualize benchmark performance across UML diagram types and complexity levels for Claude Sonnet 5, Gemini 3.1 Pro Preview, and GPT-4.1, respectively. The heatmaps provide a detailed view of model specialization and demonstrate that benchmark performance varies across both diagram notation and complexity.

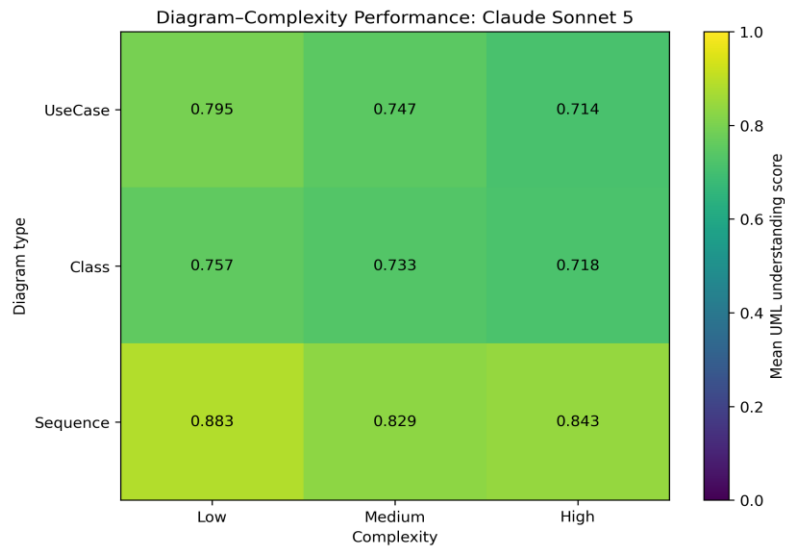


Figure C7a. Claude Sonnet 5 performance by diagram type and complexity.

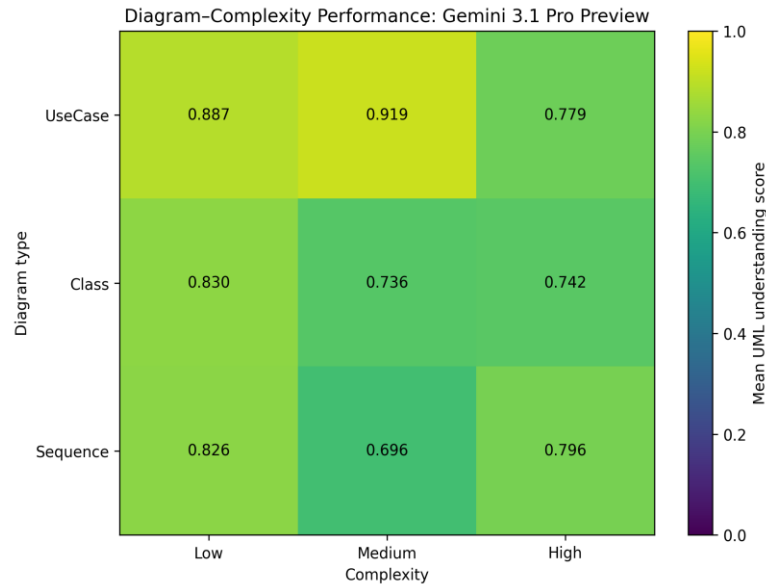


Figure C7b. Gemini 3.1 Pro Preview performance by diagram type and complexity.

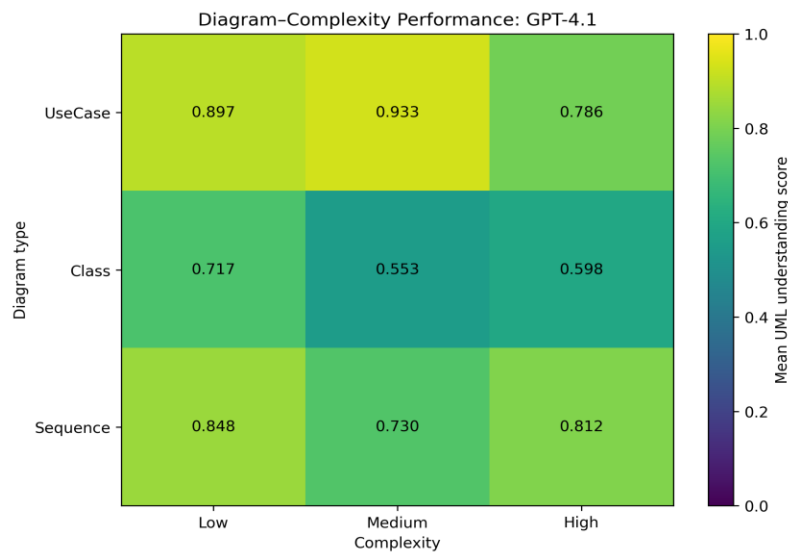


Figure C7c. GPT-4.1 performance by diagram type and complexity.

C.8 Representation Quality Diagnostics

We observed complete required-field presence for Claude Sonnet 5 and GPT-4.1 and near-complete performance for Gemini. These results complement the schema-validity findings reported under RQ5. Figure C8 reports required-field completeness across the evaluated models. Together with the schema-validation results reported in Section 4, these findings confirm that all three models consistently generated machine-processable benchmark outputs suitable for automated evaluation, with only minor schema-conformance issues observed for Gemini 3.1 Pro Preview.

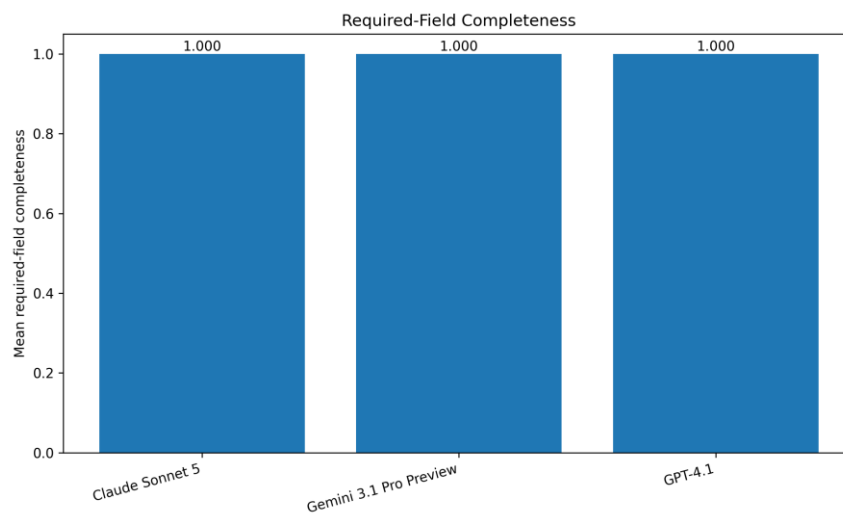


Figure C8. Mean required-field completeness by model.

APPENDIX D

Diagram-Complexity Classification Rubrics

To ensure reproducible benchmark construction, we assigned complexity using diagram-specific operational criteria rather than subjective judgment. Although all three UML diagram types represent the same software scenario, each diagram type contains different structural and behavioral characteristics.

The criteria presented below served as benchmark construction guidelines rather than absolute thresholds. Subject-matter experts considered both the number of modeling elements and the richness of their structural or behavioral relationships when assigning the final complexity level. Tables D1–D3 summarize the operational criteria for use case, class, and sequence diagrams, respectively.

Complexity	Actors	Use Cases	Relationships	Modeling Characteristics
Low	1–2	2–4	Primarily simple associations	Simple actor-use case interactions with little or no include, extend, or generalization relationships
Medium	2–4	5–8	Multiple association types	Includes include, extend, or limited generalization relationships
High	≥4	≥9	Numerous interconnected relationships	Multiple include, extend, and generalization relationships spanning several actors

Table D1. Complexity classification rubric for Use Case diagrams

Complexity	Classes	Attributes / Operations	Relationships	Modeling Characteristics
Low	2–4	Limited	Simple associations	Minimal structural complexity without inheritance or multiplicities
Medium	5–8	Moderate	Multiple relationship types	Includes inheritance, aggregation, composition, or multiplicities
High	≥9	Extensive	Dense interconnected relationships	Multiple inheritance hierarchies, multiplicities, and rich class interactions

Table D2. Complexity classification rubric for Class diagrams

Complexity	Participants	Messages	Combined Fragments	Behavioral Characteristics
Low	2–3	3–6	None	Linear interaction sequence
Medium	3–5	7–12	One fragment	Alternative or iterative execution paths
High	≥5	≥13	Multiple or nested fragments	Complex interactions involving multiple behavioral constructs

Table D3. Complexity classification rubric for Sequence diagrams

APPENDIX E

SUBJECT-MATTER-EXPERT DIAGRAM EVALUATION ANALYSIS

E.1 Scope, Units, and Analytic Procedure

One subject-matter expert (SME) created the 30 benchmark scenarios and their 90 rendered UML diagrams; two additional SMEs completed the evaluation worksheets analyzed here. Each reviewing SME rated all 90 diagrams—30 Use Case, 30 Class, and 30 Sequence diagrams—on the same 12 five-point rubric criteria. The diagram is the evaluation unit. The spreadsheet's Average Quality Score is the arithmetic mean of the 12 criterion ratings for that diagram.

The analysis includes data-completeness and formula checks; reviewer-specific descriptive statistics; internal consistency across the 12 rubric items (Cronbach's α); Pearson and Spearman association; two-way absolute-agreement and consistency intraclass correlations [ICC(A,1) and ICC(C,1)]; exact and within-one-point agreement; quadratic-weighted Cohen's κ for ordinal criteria; Cohen's κ for retain/revise/remove decisions; Bland-Altman bias and limits of agreement; diagram-type and assigned-complexity summaries; and a scenario-aggregated sensitivity analysis. Because these records comprise a census of the benchmark rather than a probability sample, significance tests are reported as diagnostics of score separation, not as population-generalization claims. With only two reviewing SMEs, reliability estimates should also be interpreted cautiously. Table E1 summarizes evaluation coverage, data integrity, and overall reviewer scores.

The Gold-Standard Interpretability criterion captures how confidently a reviewer could derive an unambiguous answer key from a rendered diagram. It does not, by itself, establish that the reviewers independently revalidated every canonical UML element or every functional requirement. In addition, domain metadata were not compared statistically because the sheets used different label granularities (for example, a specific system context in one sheet and a broader application domain in the other). Diagram ID, type, assigned complexity, and scenario suffix were fully aligned and were used for paired analyses.

Measure	Reviewer 1	Reviewer 2
Diagrams rated	90	90
Completed criterion ratings	1,080/1,080	1,080/1,080
Missing criterion ratings	0	0
Average-formula discrepancies	0	0
Retain recommendations	69	73
Revise recommendations	21	17
Remove recommendations	0	0
Overall mean (SD)	4.35 (0.48)	4.50 (0.47)
Median (IQR)	4.42 (0.75)	4.58 (0.81)
Observed range	3.25–5.00	3.50–5.00
Cronbach's α (12 criteria)	.82	.90

Table E1. Evaluation coverage, integrity checks, and overall score summary

E.2 Overall and Criterion-Level Quality

Reviewer 1's mean diagram-average score was 4.35 (SD = 0.48), and Reviewer 2's was 4.50 (SD = 0.47). Reviewer 2 scored diagrams 0.15 points higher on average; the paired Wilcoxon diagnostic was significant ($W = 851$, $p < .001$). This small shift should be interpreted as a difference in rating stringency, not evidence that one reviewer was more accurate. Internal consistency was good for Reviewer 1 ($\alpha = .82$) and high for Reviewer 2 ($\alpha = .90$), indicating that the 12 criteria generally moved together within each worksheet.

Across both reviewers, the highest pooled criterion means were Logical Relationship Correctness (4.84), Content Coherence (4.77), Educational Suitability (4.69), Complexity-Level Match (4.67), UML Notation Correctness (4.63), and Gold-Standard Interpretability (4.58). The lowest pooled means were Visual Clutter/Manageability (3.97), Spacing and Alignment (4.05), Arrow/Line Clarity (4.12), Layout Readability (4.22), and Overall Diagram Quality (4.22). The pattern indicates that the principal improvement opportunities are visual-density and routing concerns rather than the conceptual coherence or instructional suitability of the diagrams. Figure E1 compares criterion means and confidence intervals, and Table E2 reports criterion-level descriptive statistics.

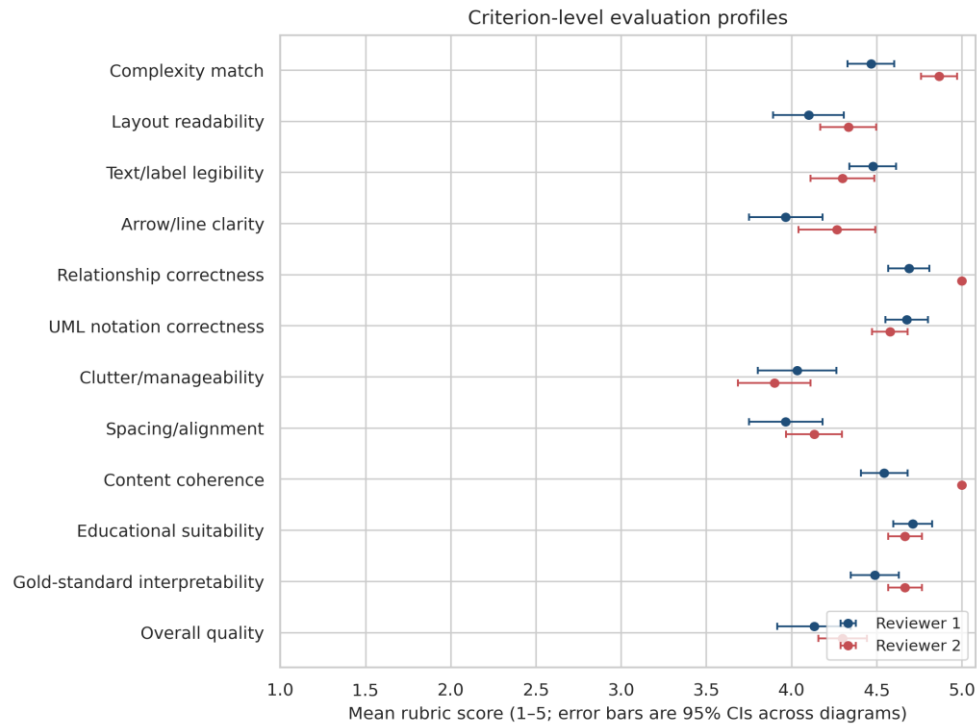


Figure E1. Criterion-level mean scores for the two reviewing SMEs. Error bars show 95% confidence intervals across the 90 diagrams and are descriptive of benchmark variation.

Criterion	Reviewer 1 mean (SD)	Reviewer 2 mean (SD)	Pooled mean	Δ R2-R1
Complexity match	4.47 (0.66)	4.87 (0.50)	4.67	+0.40
Layout readability	4.10 (0.99)	4.33 (0.78)	4.22	+0.23
Text/label legibility	4.48 (0.66)	4.30 (0.89)	4.39	-0.18
Arrow/line clarity	3.97 (1.03)	4.27 (1.08)	4.12	+0.30
Relationship correctness	4.69 (0.57)	5.00 (0.00)	4.84	+0.31
UML notation correctness	4.68 (0.60)	4.58 (0.50)	4.63	-0.10
Clutter/manageability	4.03 (1.11)	3.90 (1.02)	3.97	-0.13
Spacing/alignment	3.97 (1.03)	4.13 (0.78)	4.05	+0.17
Content coherence	4.54 (0.66)	5.00 (0.00)	4.77	+0.46
Educational suitability	4.71 (0.55)	4.67 (0.47)	4.69	-0.04
Gold-standard interpretability	4.49 (0.67)	4.67 (0.47)	4.58	+0.18
Overall quality	4.13 (1.03)	4.30 (0.68)	4.22	+0.17

Table E2. Criterion-level descriptive statistics

E.3 Diagram Type and Assigned Complexity

Diagram type did not produce a detectable score difference for either reviewer (Reviewer 1: $H = 0.06$, $p = 0.969$; Reviewer 2: $H = 4.34$, $p = 0.114$). Assigned complexity produced a pronounced separation (Reviewer 1: $H = 70.16$, $p < .001$, $\epsilon^2 = 0.78$; Reviewer 2: $H = 76.46$, $p < .001$, $\epsilon^2 = 0.86$). Mean scores decreased from Low (4.82 and 4.94) to Medium (4.46 and 4.64) to High (3.78 and 3.93), which indicates that the rubric detected the intended rise in visual and relational difficulty. Table E3 reports the corresponding means, and Figures E2 and E3 show the cross-tabulated means and score distributions.

Subgroup	n per SME	Reviewer 1 mean (SD)	Reviewer 2 mean (SD)	Paired mean
All diagrams	90	4.35 (0.48)	4.50 (0.47)	4.43
Use Case	30	4.36 (0.49)	4.39 (0.54)	4.38
Class	30	4.36 (0.45)	4.62 (0.44)	4.49
Sequence	30	4.34 (0.51)	4.49 (0.40)	4.41
Low	30	4.82 (0.11)	4.94 (0.07)	4.88
Medium	30	4.46 (0.25)	4.64 (0.19)	4.55
High	30	3.78 (0.23)	3.92 (0.25)	3.85

Table E3. Diagram-average quality by diagram type and assigned complexity

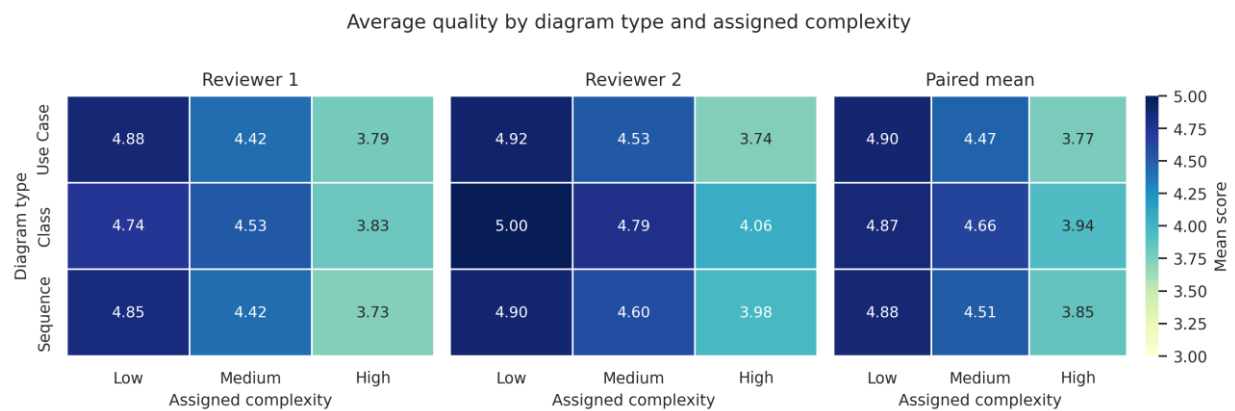


Figure E2. Mean diagram quality by UML diagram type and assigned complexity. Each heatmap cell contains 10 diagrams per reviewer.

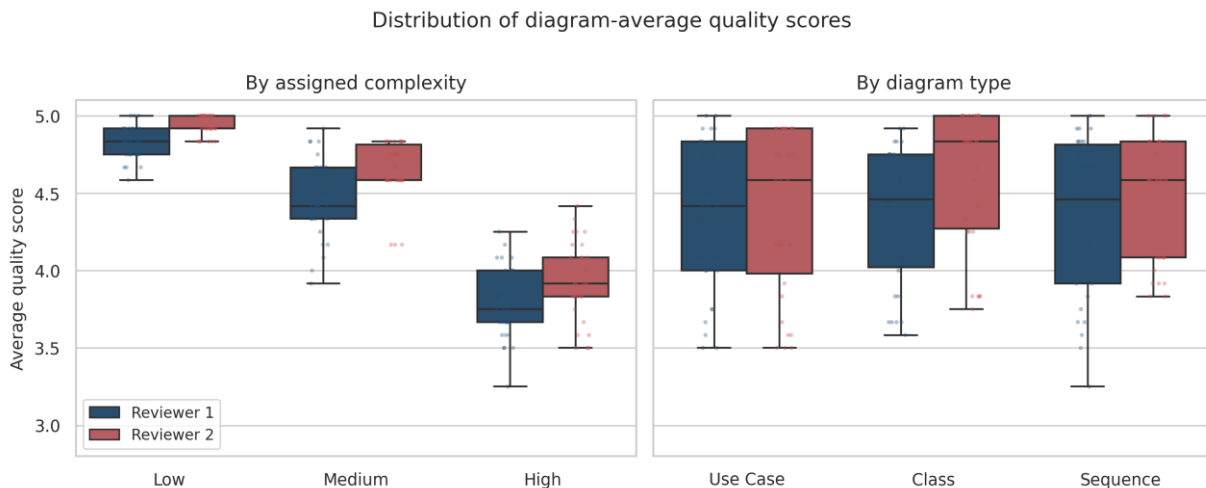


Figure E3. Distribution of diagram-average quality scores by assigned complexity and diagram type.

E.4 Diagram-Level Inter-Reviewer Agreement

Diagram-average scores were strongly associated (Pearson $r = 0.84$; Spearman $\rho = 0.83$; both $p < .001$). Absolute agreement was $ICC(A,1) = 0.81$, with a bootstrap 95% interval of 0.74–0.86; consistency was $ICC(C,1) = 0.84$ (0.79–0.89). Exact equality of the 12-item averages occurred for 12/90 diagrams, 66.7% were within 0.25 points, and 92.2% were within 0.50 points. Figure E4 plots the paired reviewer scores.

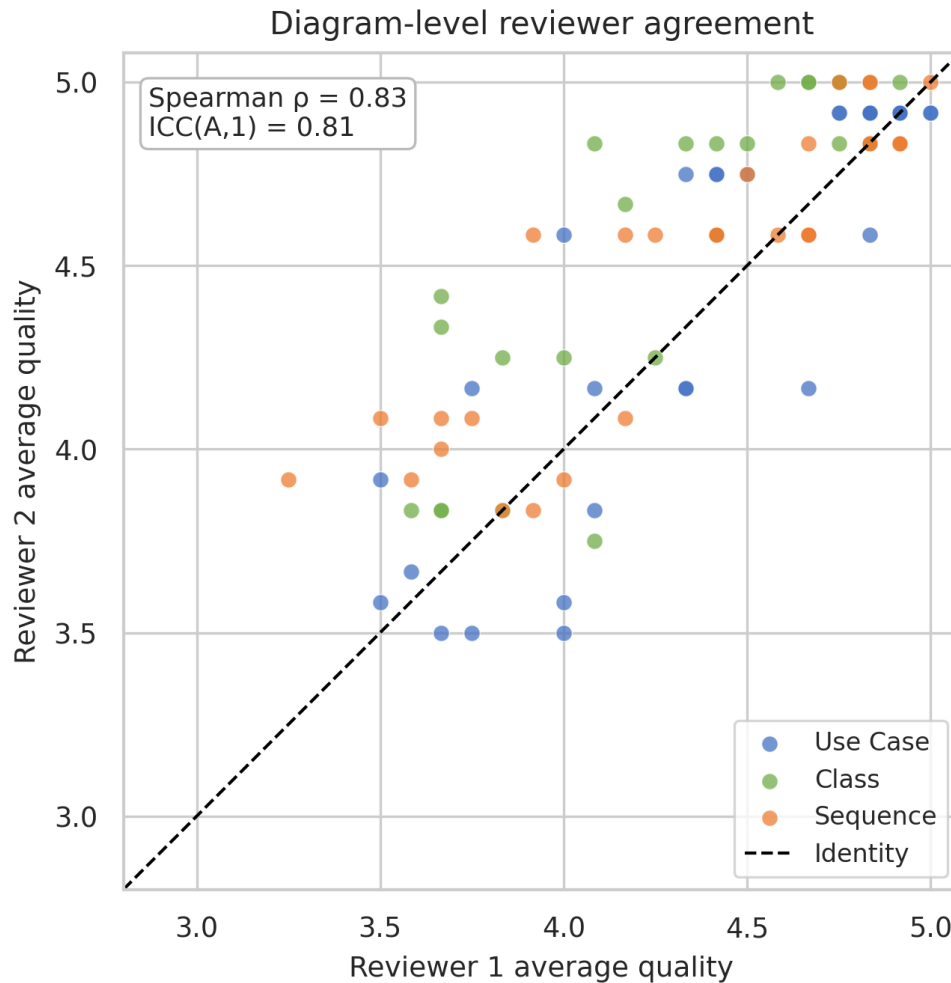


Figure E4. Reviewer 1 versus Reviewer 2 diagram-average scores. The dashed identity line represents exact equality.

The Bland–Altman mean difference was +0.15 points (Reviewer 2 minus Reviewer 1), with 95% limits of agreement from -0.37 to +0.67. Thus, Reviewer 2 was modestly more generous on average, but most pairwise differences were small relative to the five-point scale. The plot does not show a clear increase in disagreement as average quality rises; ceiling compression is visible near 5.0. Figure E5 presents the corresponding Bland–Altman analysis.

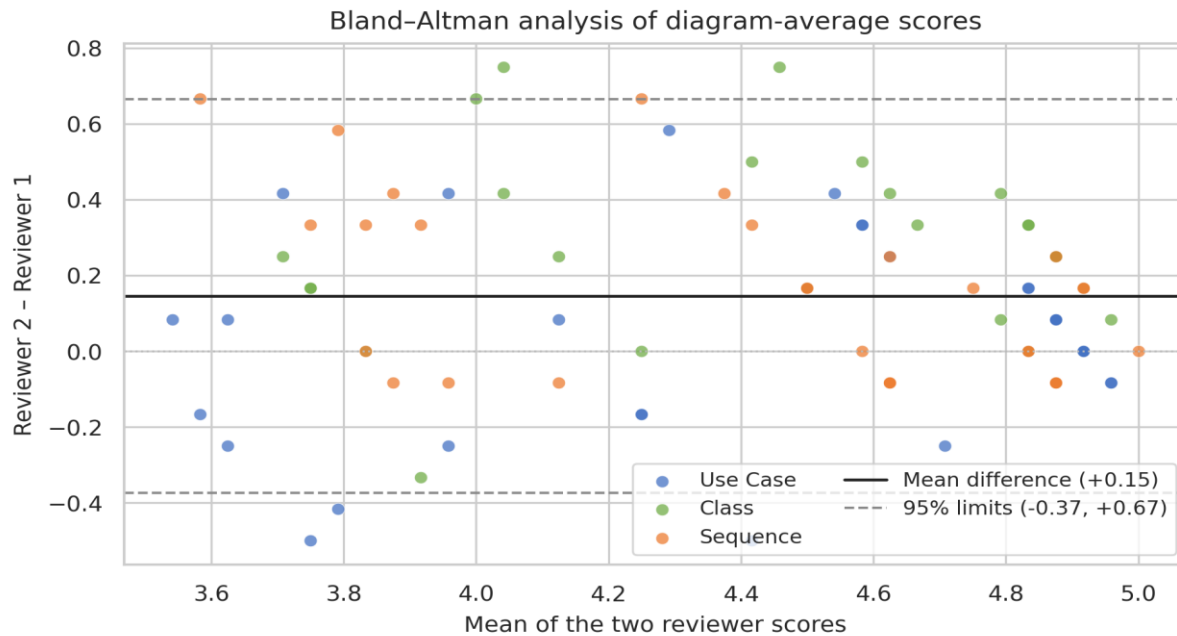


Figure E5. Bland-Altman assessment of reviewer bias and dispersion for diagram-average scores.

E.5 Criterion-Level Agreement and Bounded-Scale Effects

Exact agreement on individual criteria ranged from 44.4% to 74.4%, while agreement within one scale point ranged from 81.1% to 98.9%. Weighted κ and ICC were strongest for Layout Readability, Visual Clutter/Manageability, Overall Diagram Quality, Spacing and Alignment, Arrow/Line Clarity, and Text/Label Legibility. Chance-corrected coefficients were low or undefined in practical terms for several ceiling-dominated criteria. In particular, Reviewer 2 assigned a score of 5 to every diagram for Logical Relationship Correctness and Content Coherence; the absence of variance forces κ and ICC toward zero even though 94.4% and 93.3% of ratings, respectively, were within one point. This is the bounded-scale/prevalence effect: high observed agreement can coexist with a low chance-corrected coefficient when ratings are concentrated in one category and contain many ties. Table E4 reports the criterion-level agreement coefficients, and Figure E6 contrasts chance-corrected with observed agreement.

Criterion	Exact %	Within 1 %	Weighted κ	ICC(A,1)	Holm p
Complexity match	51.1	86.7	-0.06	-0.06	< .001
Layout readability	56.7	93.3	0.61	0.61	= .086
Text/label legibility	56.7	92.2	0.47	0.47	= .571
Arrow/line clarity	46.7	81.1	0.47	0.48	= .013
Relationship correctness	74.4	94.4	0.00	0.00	< .001
UML notation correctness	57.8	98.9	0.25	0.25	= .586
Clutter/manageability	50.0	90.0	0.60	0.60	= .586
Spacing/alignment	44.4	93.3	0.52	0.52	= .586
Content coherence	62.2	93.3	0.00	0.00	< .001
Educational suitability	65.6	96.7	0.14	0.14	= .586
Gold-standard interpretability	58.9	94.4	0.18	0.18	= .359
Overall quality	53.3	92.2	0.54	0.55	= .586

Table E4. Criterion-level inter-reviewer agreement

Note. Holm-adjusted p values refer to paired Wilcoxon diagnostics of reviewer score distributions, not tests of whether a diagram is acceptable. Negative or zero chance-corrected coefficients should be read together with observed agreement because bounded scores, ceiling effects, and tied observations can make κ and ICC unstable.

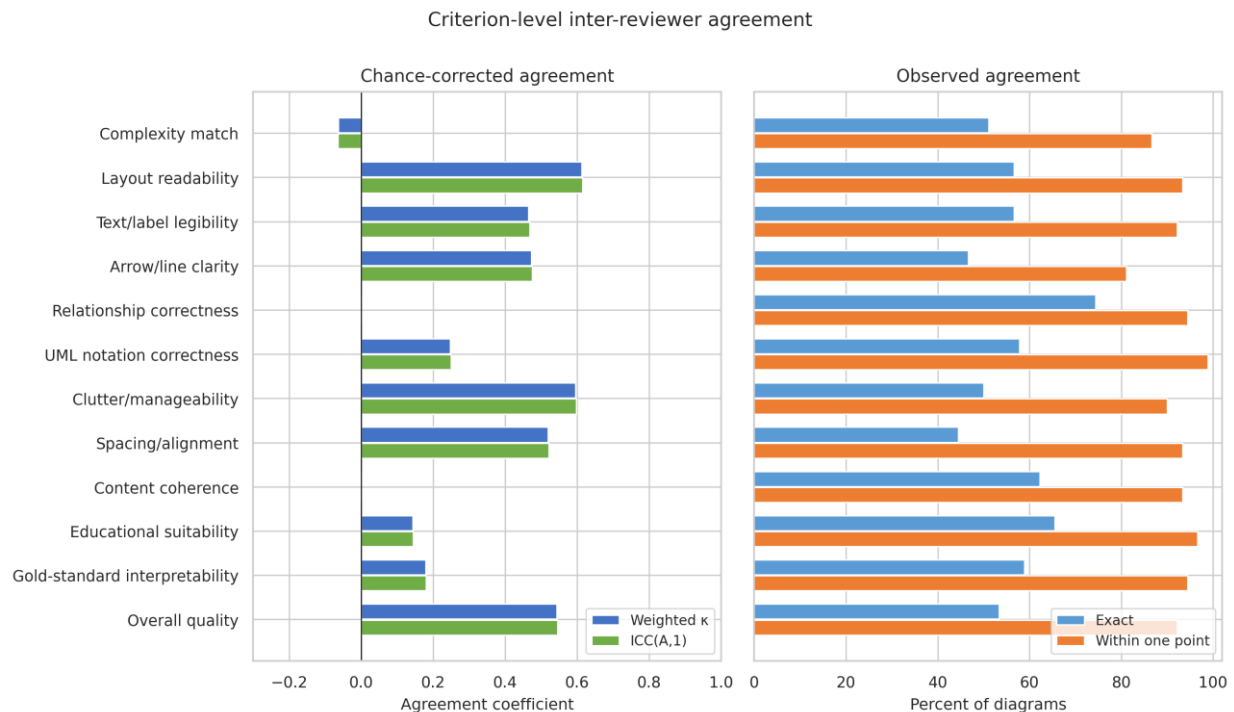


Figure E6. Criterion-level chance-corrected and observed inter-reviewer agreement.

E.6 Complexity-Classification Validation

Each reviewer's perceived complexity matched the assigned label for 84/90 diagrams (93.3%; weighted $\kappa = 0.95$). More importantly, the two reviewers' perceived labels agreed on all 90 diagrams (weighted $\kappa = 1.0$). The same six diagrams—SQ-L03, SQ-L04, SQ-L06, SQ-L07, SQ-L09, and SQ-L10—were assigned Low but perceived as Medium by both SMEs. This unanimous mismatch is actionable evidence that these six Sequence diagrams should either be reclassified as Medium or accompanied by a clearer operational explanation for retaining the Low designation.

Complexity classification alignment (counts)

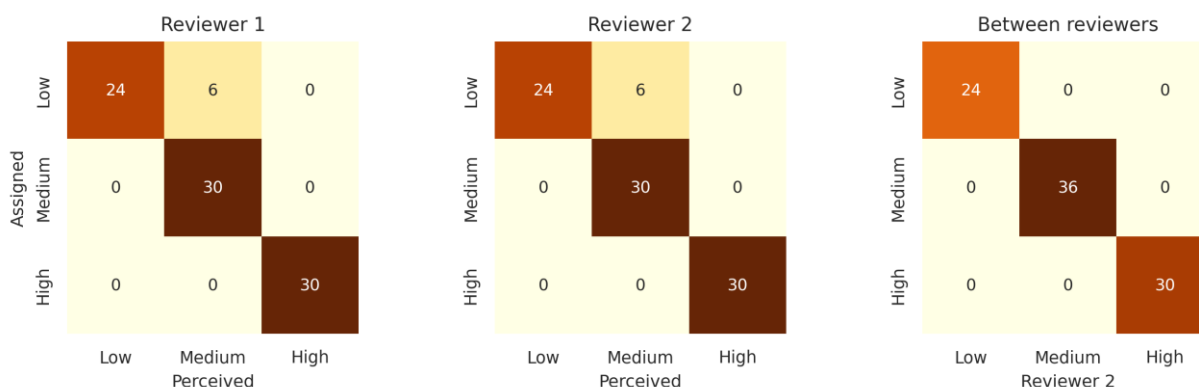


Figure E7. Assigned-versus-perceived complexity and between-reviewer complexity alignment.

E.7 Retain/Revise Decisions and Issue Patterns

Reviewer 1 recommended retaining 69 diagrams and revising 21; Reviewer 2 recommended retaining 73 and revising 17. Neither reviewer recommended removal. The action recommendations matched for 76/90 diagrams (84.4%; Cohen's $\kappa = 0.53$). The joint decisions were 64 both retain, 12 both revise, and 14 mixed retain/revise. Every revision recommendation occurred among the 30 High-complexity diagrams: the reviewers jointly retained 4, jointly revised 12, and disagreed on 14. All 60 Low- and Medium-complexity diagrams were retained by both SMEs. Figure E8 presents the recommendation proportions and action matrix.

Retain/revise/remove decisions

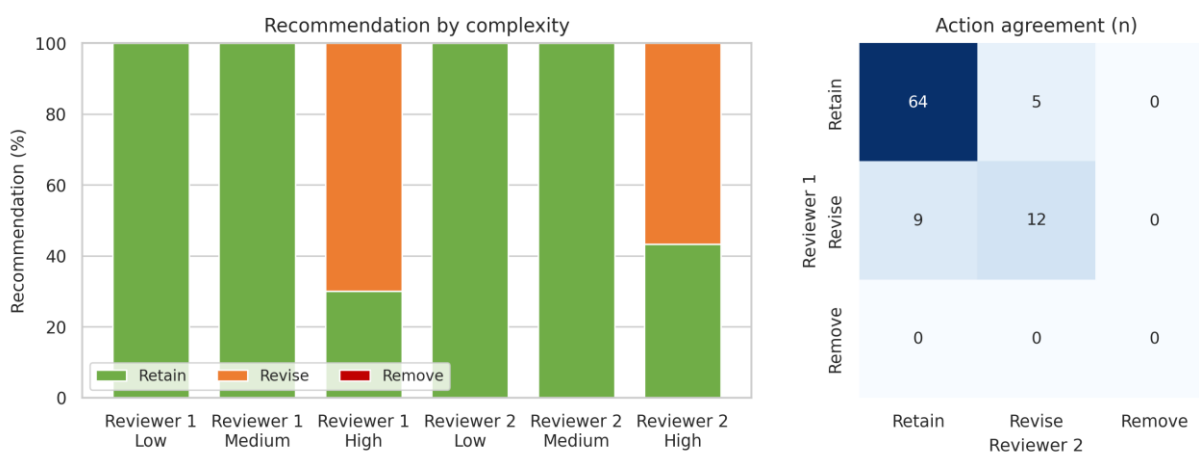


Figure E8. Recommendation proportions by complexity and the reviewer action confusion matrix.

Primary-issue fields were used differently. Reviewer 1 recorded no primary issue for 69 diagrams and mainly used Layout/Spacing/Alignment (18) or Visual Clutter/Density (3) when a change was warranted.

Reviewer 2 recorded minor issues even for many retained diagrams and distributed them across Visual Clutter/Density (18), Text/Label Legibility (17), Arrow/Line Clarity (16), Complexity Classification (6), and Layout/Spacing/Alignment (1). These frequencies therefore describe each reviewer's documentation practice and should not be interpreted as direct category-by-category agreement. They nevertheless converge on a common design concern: dense high-complexity diagrams need improved spacing, routing, and legibility. Figure E9 summarizes the normalized issue categories.

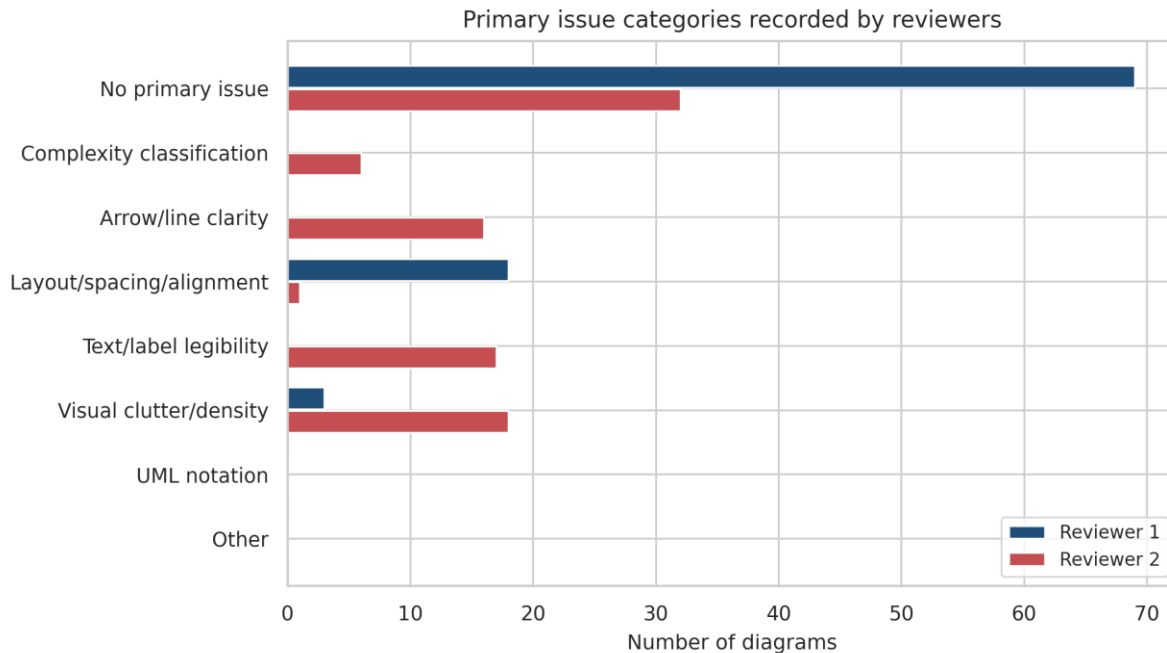


Figure E9. Normalized primary-issue categories recorded in the two worksheets.

E.8 Scenario-Aggregated Sensitivity Analysis

Each scenario key contributes three notation-specific views (Use Case, Class, and Sequence). Averaging those three diagram scores for each of the 30 scenario keys produced still stronger reviewer agreement: Pearson $r = 0.94$, Spearman $\rho = 0.89$, and ICC(A,1) = 0.89. This sensitivity analysis shows that reviewer conclusions remain aligned after scenario-level aggregation and reduces the influence of any single notation-specific score.

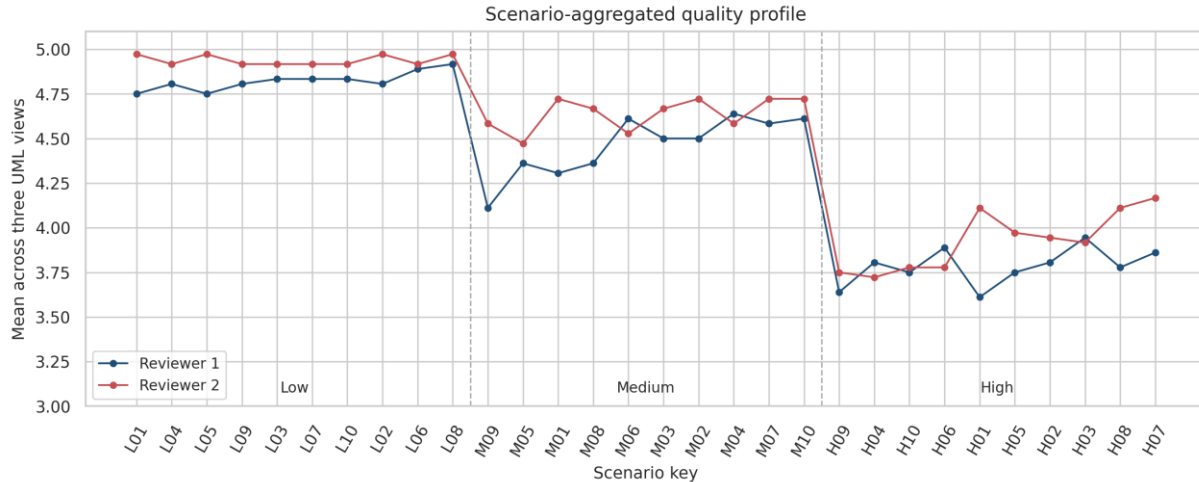


Figure E10. Reviewer quality profiles after averaging the three UML views within each scenario key. Scenario keys are ordered within each complexity stratum by paired mean quality.

E.9 Diagram-Level Revision Priorities

The 15 lowest paired mean scores all belonged to High-complexity diagrams. Their weakest cells cluster in Layout Readability, Arrow/Line Clarity, Visual Clutter/Manageability, Spacing and Alignment, and Overall Diagram Quality. By contrast, Logical Relationship Correctness, Content Coherence, Educational Suitability, and Gold-Standard Interpretability generally remained high. This pattern supports targeted visual refactoring rather than wholesale replacement of scenario content. Figure E11 visualizes these criterion patterns; Table E5 lists diagrams jointly recommended for revision, and Table E6 reports the largest reviewer-score differences.

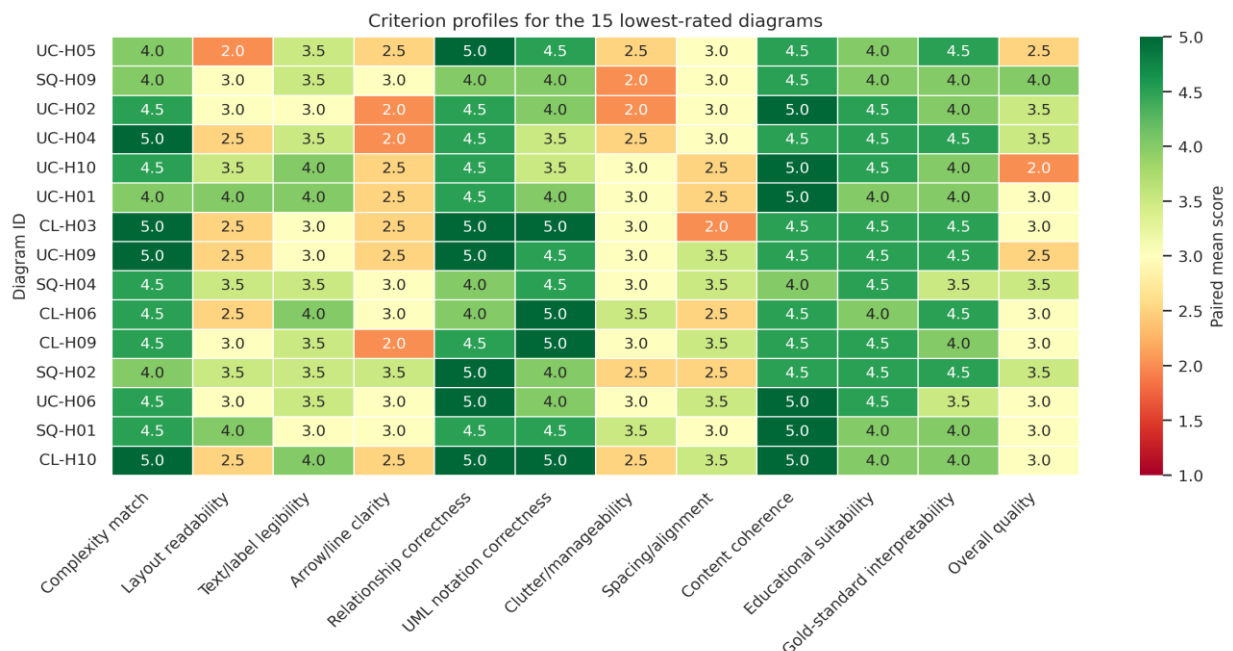


Figure E11. Paired criterion means for the 15 lowest-rated diagrams.

Diagram ID	Type	Paired mean	Lowest paired criterion	Score
UC-H05	Use Case	3.54	Layout readability	2.0
SQ-H09	Sequence	3.58	Clutter/manageability	2.0
UC-H02	Use Case	3.58	Arrow/line clarity	2.0
UC-H04	Use Case	3.62	Arrow/line clarity	2.0
UC-H10	Use Case	3.62	Overall quality	2.0
CL-H03	Class	3.71	Spacing/alignment	2.0
SQ-H04	Sequence	3.75	Arrow/line clarity	3.0
CL-H06	Class	3.75	Layout readability	2.5
CL-H09	Class	3.75	Arrow/line clarity	2.0
CL-H10	Class	3.83	Layout readability	2.5
SQ-H10	Sequence	3.83	Spacing/alignment	2.5
SQ-H08	Sequence	3.88	Text/label legibility	3.0

Table E5. Diagrams jointly recommended for revision

Diagram ID	Type	Complexity	R1	R2	R2-R1	Δ	Action Pair (R1/R2)
CL-H01	Class	High	3.67	4.42	+0.75	0.75	Revise / Retain
CL-M01	Class	Medium	4.08	4.83	+0.75	0.75	Retain / Retain
CL-H08	Class	High	3.67	4.33	+0.67	0.67	Revise / Retain
SQ-H09	Sequence	High	3.25	3.92	+0.67	0.67	Revise / Revise
SQ-M09	Sequence	Medium	3.92	4.58	+0.67	0.67	Retain / Retain
SQ-H02	Sequence	High	3.50	4.08	+0.58	0.58	Revise / Retain
UC-M09	Use Case	Medium	4.00	4.58	+0.58	0.58	Retain / Retain
CL-M02	Class	Medium	4.33	4.83	+0.50	0.50	Retain / Retain
CL-M05	Class	Medium	4.17	4.67	+0.50	0.50	Retain / Retain
UC-H09	Use Case	High	4.00	3.50	-0.50	0.50	Retain / Revise
UC-M04	Use Case	Medium	4.67	4.17	-0.50	0.50	Retain / Retain

Table E6. Largest absolute reviewer differences in diagram-average scores, including ties at the tenth-ranked cutoff

Note. R1 = Reviewer 1; R2 = Reviewer 2; Δ = R2 - R1. Action Pair reports Reviewer 1's recommendation followed by Reviewer 2's recommendation. All diagrams tied at the top-ten cutoff ($|\Delta| = 0.50$) are included; therefore, the table contains 11 diagrams.

E.10 Limitations

The analysis is based on two reviewing SMEs, so reliability estimates do not characterize the full population of potential expert reviewers. Scores are ordinal, bounded, and frequently tied; parametric summaries are included for interpretability but are accompanied by rank-based and agreement diagnostics. Reviewer granularity also differed: Reviewer 1 used 82 distinct 12-item score profiles, whereas Reviewer 2 used 27, with a largest repeated profile of 14 diagrams. This difference increases ties and contributes to the prevalence sensitivity of κ and ICC. Finally, the rubric validates perceived diagram quality, notation, coherence, educational suitability, and answer-key interpretability; it should not be described as a full independent re-derivation of every canonical element or functional requirement.