Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 4, August 2026, pp. 3268~3279
ISSN: 2302-9285, DOI: 10.11591/eei.v15i4.11226

Toward generalizable unified modeling language automation: a dual case study on class and use case diagram generation

Van-Viet Nguyen[1], Huu-Khanh Nguyen[1,2], Kim-Son Nguyen[1], Thi Minh-Hue Luong[1], Duc-Quang Vu[1], The-Vinh Nguyen[1]
[1]Faculty of Information Technology, Thai Nguyen University of Information and Communication Technology, Thai Nguyen, Vietnam
[2]Distance Learning Center, Thai Nguyen University, Thai Nguyen, Vietnam

Article Info

Article history:

Received Aug 18, 2025
Revised May 26, 2026
Accepted Jun 19, 2026

Keywords:

Automated unified modeling language generation
AuUMLCode
Model reasoning
PlantUML
Software engineering

ABSTRACT

Manual generation of unified modeling language (UML) diagrams creates bottlenecks in agile development due to inconsistency and labor intensity. While large language models (LLMs) offer generative capabilities, existing solutions suffer from data scarcity and inadequate evaluation tools. To address this, we present a dual-LLM pipeline integrating lightweight specification generation with reasoning-oriented code synthesis. Uniquely, this framework employs a weighted multimodal validation module utilizing diverse vision-language models (VLMs) to assess diagrammatic fidelity. We further address the data shortage by releasing benchmark datasets comprising 5,000 class and 3,000 use case diagrams. Empirical results demonstrate a 95.8% rendering success rate for class diagrams and strong semantic alignment for use case models. By mitigating structural and behavioral reasoning conflicts, this research provides a replicable architecture and rigorous assessment methodology, establishing a robust foundation for scalable, artificial intelligence-driven automation in software engineering.

Corresponding Author:

The-Vinh Nguyen
Faculty of Information Technology
Thai Nguyen University of Information and Communication Technology
Quyet Thang Ward, Thai Nguyen 25000, Vietnam
Email: vinhnt@ictu.edu.vn

1. INTRODUCTION

In software engineering, effective communication among stakeholders requires a common language regardless of their expertise. Unlike English (which requires learning a large vocabulary and grammar), unified modeling language (UML) uses a compact set of visual notations that convey complex system structures and behaviors through diagrams [1]. As a result, UML has become a standard for modeling software-intensive systems. Despite its widespread adoption, UML diagrams are still mostly created manually. This process is labor-intensive, error-prone, and poorly suited to modern Agile and DevOps environments, where system requirements change frequently [2], [3]. When design documentation fails to keep pace with system implementation, documentation mismatches occur, gradually creating engineering debt, and reducing design reliability [3], [4]. For this reason, maintaining consistency between requirements, design models, and codebases continues to pose a major challenge in large-scale software development.

Automating the construction of UML models from written requirements specifications has long been considered a daunting task. The main reason is that natural language in the specification inherently contains many ambiguous and dynamic elements [5]. Early work to alleviate this issue relied rule-based natural language processing (NLP) techniques and manually designed heuristic sets to extract UML

components from text [6], [7]. While these methods performed relatively well in tightly constrained contexts, they lacked robustness when dealing with informal, incomplete, or unstructured specifications. This limited their applicability in real-world scenarios. The emergence of large language models (LLMs) has brought about significant changes in this field [8], [9]. Transformer-based models such as LLaMA and DeepSeek have demonstrated the ability to function as "novice analysts," translating natural language descriptions into intermediate modeling representations with minimal supervision [10], [11]. Empirical studies further showed that instruction-based training substantially improves the capture of architectural intent from textual inputs [12]. In parallel, advances in vision–language models (VLMs), including Qwen2-VL and Aya-Vision, have enabled machines to reason directly over visual artifacts, opening new possibilities for automated validation of software diagrams [13], [14].

However, fully automated UML generation remains underexplored and fragmented. Existing studies typically focus on isolated diagram types or narrow syntax translations, without addressing the complete pipeline from requirements to validated design artifacts [15]. Progress is further constrained by the lack of large-scale, high-quality datasets containing semantically validated UML diagrams; available datasets often exhibit duplication, limited diversity, or missing ground truth annotations [16], [17]. Most critically, current evaluation practices rely either on text-based metrics such as BLEU or ROUGE, or on subjective manual inspection both of which fail to capture the structural logic and semantic correctness of visual models [18]–[20]. Prior work consistently reports that LLMs struggle with the strict syntax and spatial constraints of UML without specialized feedback or fine-tuning mechanisms [21], [22].

To address these limitations, this paper introduces an end-to-end framework for automated UML generation and validation based on a dual-LLM architecture and a multimodal weighted evaluation strategy. The generation process is decomposed into two stages: lightweight specification extraction using LLaMA-3.2, followed by reasoning-oriented UML synthesis using DeepSeek-R1. This separation resolves the tension between creative language generation and strict syntactic adherence. To mitigate data scarcity, we further release two large-scale benchmark datasets comprising 5,000 class diagrams and 3,000 use case diagrams derived from realistic software scenarios.

The contributions of this study are threefold. First, it introduces a Dual-LLM architecture featuring a two-stage pipeline that separates specification generation from UML synthesis, thereby ensuring both computational efficiency and semantic accuracy. Second, the study proposes a multimodal weighted validation framework that aggregates insights from diverse VLMs specifically Qwen2.5-VL, LLaMA-Vision, and Aya-Vision to objectively assess diagrammatic fidelity in a manner that surpasses traditional metrics. Finally, this work provides the research community with a scalable resource through the release of validated, large-scale benchmark datasets for both class and use case diagrams.

2. RELATED WORK

Early works focused on syntactic transformations from textual requirements into UML diagrams using handcrafted grammars, patterns, and linguistic heuristics. These approaches mapped noun verb phrases to UML elements such as classes, attributes, and relationships [23]. While effective for narrow domains, they lacked scalability and generalization when applied to heterogeneous requirements.

With the advent of deep learning, sequence-to-sequence (seq2seq) architectures were applied to map natural language descriptions into intermediate specification languages [24], using NLP techniques, heuristics [19], [25]. These methods improved adaptability compared to rule-based systems but were constrained by data scarcity and the inability to handle long-range reasoning in complex software descriptions.

Recent progress in transformer-based models such as GPT [26], LlaMA [27], [28] has enabled direct text-to-code and text-to-UML generation. Yang and Sahraoui [23] explored automated extraction of UML class diagrams from unstructured natural language specifications, while Conrardy and Cabot [29] investigated image-to-UML generation using multimodal LLMs, model generation with LLMs [4], automatic code generation [30], image based UML diagram generation using LLMs [29], extracting UML models from images [31]. These studies highlight the potential of LLMs in bridging unstructured requirements with formalized UML artifacts. However, they primarily focus on single diagram types and lack standardized validation mechanisms.

In parallel, vision-language representation learning has progressed significantly with models such as UNITER [32] and VinVL [33], generation from diagram images using multimodal LLMs [15], which enable joint reasoning over text and images, semantic recognition from images [34]. More recent efforts Qwen2-VL [13], [35], and Aya-Vision [14] extend these capabilities to multilingual and high-resolution reasoning, providing opportunities for evaluating structured visual artifacts like UML diagrams.

Despite these advances, two persistent challenges remain: i) the scarcity of large, high-quality datasets for UML generation and validation and ii) the absence of robust evaluation frameworks that capture

both structural fidelity and semantic alignment. Existing metrics (BLEU and ROUGE) [36], [37] are inadequate for structured diagram evaluation, while manual assessments hinder reproducibility.

The paradigm of automated modeling has transitioned from rule-based extraction to generative synthesis. Comprehensive surveys by Ahmed *et al.* [3] and Abdelnabi *et al.* [38] indicate that while NLP-based transformation rules provide determinism, they lack the flexibility to handle ambiguous natural language requirements. To address this, recent approaches have leveraged the reasoning capabilities of advanced LLMs. The DeepSeek-R1 model utilizes reinforcement learning to incentivize reasoning paths, showing promise in solving complex logic tasks inherent in system design [39]. Concurrently, Bates *et al.* [15] explored multimodal LLMs for generating code from diagram images, suggesting that the "chain-of-thought" prompting method [40] significantly enhances the structural correctness of the output. These studies collectively suggest that decomposing the generation process into specification extraction and code synthesis a dual step approach may yield superior results compared to direct end-to-end generation.

Parallel to generation, the evaluation of visual software artifacts has evolved with the advent of vision-language models (VLMs). As noted in a 2025 survey by Danish *et al.* [17], VLMs have achieved state-of-the-art performance in cross-modal understanding, making them suitable for validating diagrammatic fidelity. Benchmarking studies like MMMU [41] reveal that models such as Qwen2-VL [13] and Aya-Vision [14] possess strong optical character recognition and spatial reasoning capabilities tailored for high-resolution document analysis, a benchmark for evaluating automated UML synthesis [42]. However, the application of these models in validating UML syntax remains underexplored. Most existing evaluation frameworks rely on textual metrics (BLEU and ROUGE) or pixel-wise comparison, which fail to capture the semantic nuances of software architecture. This highlights the need for a specialized multimodal validation framework that aggregates insights from diverse VLM architectures to provide a reliable quality score.

Unlike previous studies that either focused on single diagram types or relied on subjective/manual evaluation this paper introduces three novel contributions:

- Dual-LLM pipeline: a two-stage architecture that separates lightweight specification generation from reasoning-intensive UML synthesis, ensuring both computational efficiency and semantic accuracy.
- Multimodal weighted validation: a reproducible evaluation framework that leverages multiple vision–language models (Qwen2.5-VL-3B, LLaMA-3.2-11B-Vision, and Aya-Vision-8B) and aggregates their outputs through weighted scoring guided by the MMMU benchmark, mitigating individual model bias.
- Benchmark dataset release: the construction of two large-scale, validated datasets (5,000 class diagrams and 3,000 use case diagrams), providing the community with a scalable resource for future research on UML automation.

Together, these contributions establish the first end-to-end reproducible framework that integrates requirement to UML generation with automated multimodal validation, paving the way toward generalizable UML automation in artificial intelligence (AI)-driven software engineering.

3. METHOD

Our method comprises a unified, sequential pipeline adaptable to different UML diagram types, moving from specification generation to final multimodal scoring.

3.1. Requirement specification generation

A lightweight language model, LLaMA 3.2 1B-Instruct, is optimized for instruction-following tasks [28]. The model is good at producing text, asking questions, and doing some light logical reasoning with only 1 billion parameters. This makes it resource-efficient for edge applications or situations where computing power is limited. LLaMA 3.2 1B's "Instruct" version is fine-tuned on a high-quality instruction dataset to consistently interpret input requests and provide succinct responses. LLaMA 3.2 1B-Instruct is a suitable candidate for small language model (SLM) research in the areas of code generation, NLP, and intelligent education, owing to its efficacy and performance.

We conducted a feasibility study comparing LLaMA-3.2-1B-Instruct with Gemma-2-2B and DeepSeek-R1-Distill-Qwen-1.5B. LLaMA 3.2-1B-Instruct was the most efficient model, with an inference time of 60 seconds per 100 prompts (Figure 1). DeepSeek-R1-Distill-Qwen-1.5B required 87 seconds, whereas Gemma-2-2B took 116 seconds. Its compact dimension renders it competitive in structured generation tasks such as code synthesis and UML modeling. So, we selected LLaMA-3.2-1B-Instruct as the primary model for our research due to its optimal equilibrium between speed and quality.

To ensure high-quality textual inputs, we developed a structured prompt template that guides the model to act as a product owner. This approach focuses on functional requirements while intentionally excluding technical implementation details, as illustrated in Table 1.

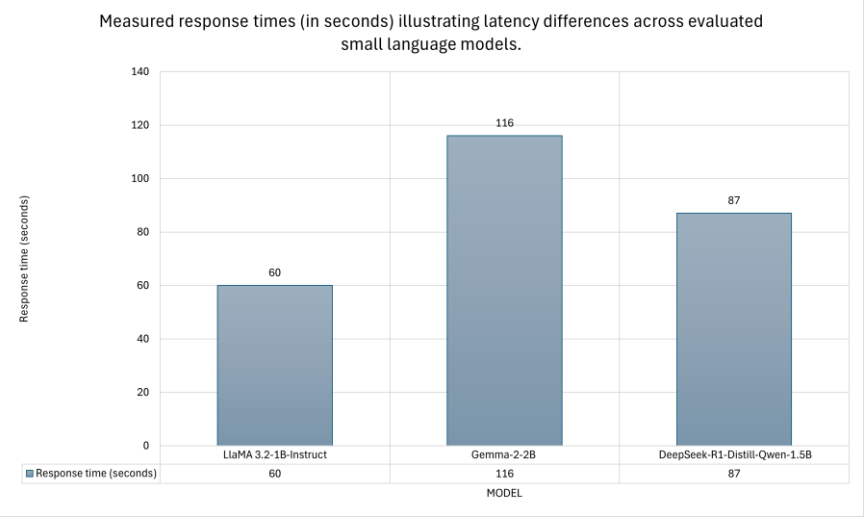

Figure 1. Measured response times (in seconds) illustrating latency differences across evaluated small language models

Table 1. Prompt templates for software requirement generation

Prompt template	LLaMA-3.2-1B output
System instruction: "Act as an end-user or product owner who has conceived a software feature. Describe the feature's functionality from a user's perspective. Focus on 'what' the system does rather than 'how' it is implemented. Avoid technical jargon, class names, or implementation details." Input: "Generate a detailed user requirement for a note-taking application feature."	"As a user, I want to be able to easily find and view all my notes in a single, organized list. I want to see a comprehensive overview, including any attachments or comments, so I can review them at a glance. Additionally, I need a search function for specific notes or keywords to quickly locate relevant information."

3.2. Unified modeling language code synthesis with reasoning

We utilize DeepSeek-R1-Distill-Qwen-32B, a model distilled from Qwen-72B to optimize reasoning and code generation while minimizing inference costs. With 32 billion parameters trained on extensive code and instruction datasets, it delivers performance comparable to larger LLMs, specifically excelling in multi-hop reasoning and complex synthesis [39]. For diagram rendering, we leverage PlantUML, an open-source tool that transforms textual descriptions into UML diagrams. Its concise, code-like syntax offers a lightweight alternative to graphical tools, facilitating seamless integration with version control (Git) and DevOps workflows [29]. By treating diagrams as code, PlantUML ensures high maintainability, reproducibility, and collaboration, making it an ideal solution for automated software modeling.

3.3. Diagram rendering and multimodal validation

We automate quality assessment using three diverse open-source VLMs: Qwen2.5-VL-3B, LLaMA-3.2-11B-Vision, and Aya-Vision-8B [13], [14], [28]. This selection integrates varying architectural perspectives ranging from efficient high resolution processing to complex visual reasoning and multilingual alignment effectively mitigating individual model bias. Their reasoning capabilities are benchmarked against MMMU to inform a robust [43], weighted scoring mechanism for objective diagram evaluation.

3.4. Weighted scoring

A defining characteristic of this algorithm is its ability to dynamically filter invalid inputs. In this system, model outputs are quantified on a scale from 0 to 6, where a score of zero designates an invalid or failed evaluation. To ensure the final metric remains robust, a logical indicator acts as a gatekeeper during the aggregation process. If a model returns a zero, it is strictly treated as non-participatory for that specific record.

Consequently, the composite score is computed by summing the weighted values of only the valid scores. Crucially, the normalization factor the divisor in the calculation adjusts dynamically for each record. It consists solely of the sum of weights from the models that provided valid feedback, rather than the total weight of all existing models. This mechanism prevents invalid zeros from artificially deflating the aggregate score. By isolating valid contributions, this method produces a normalized, reliability-adjusted consensus, ensuring a fair and accurate representation of data quality even when certain models fail to generate usable inferences.

The formula for S_j is defined as (1):

$$S_i = \frac{\sum_{i=1}^{N} \delta(s_{ij}) \cdot s_{ij} \cdot w_i}{\sum_{i}^{N} \delta(s_{ij}) \cdot w_i} \quad (1)$$

3.5. The unified generation and validation pipeline

The automated generation of UML diagrams from natural language requires a sophisticated balance between architectural reasoning and syntactic precision. To bridge the gap between high-level requirements and formal visual representations, we propose a robust multi-stage framework that leverages specialized LLMs for distinct cognitive tasks. This modular approach ensures that semantic intent is preserved through a structured pipeline of refinement, synthesis, and multimodal verification. Specifically, the framework operates in three stages:

- Stage 1 technical specification generation: LLaMA 3.2-1B-Instruct is refined to create specifications based on the type of diagram: the model plays the role of "system architect" for class diagrams to define static structures (classes, relationships), and "requirements analyst" for use case diagrams to describe functional interactions (actors, use cases).
- Stage 2 UML PlantUML code generation: the detailed specification from Stage 1 is passed to the DeepSeek-R1-Distill-Qwen-32B model. Acting as a "UML expert", the model translates the specification into syntactically valid PlantUML code. The prompt emphasizes correct syntax for the specific diagram type and encourages chain-of-thought reasoning using <think> tags to decompose the problem effectively.
- Stage 3 diagram rendering and multimodal validation: the PlantUML code is rendered as an image (syntax errors receive a score of 0) and verified by three VLMs (Qwen, LLaMA, and Aya). These models evaluate the fit between the diagram and the description on a 6-point scale, and the results are then aggregated into a common score using a weighted average mechanism. The unified generation and validation flow is presented in Figure 2 which describes the three stages operation to understand the generation and validation flow of the proposed method. To improve clarity and reproducibility, Algorithm 1 summarizes the complete dual-LLM generation and multimodal validation process.

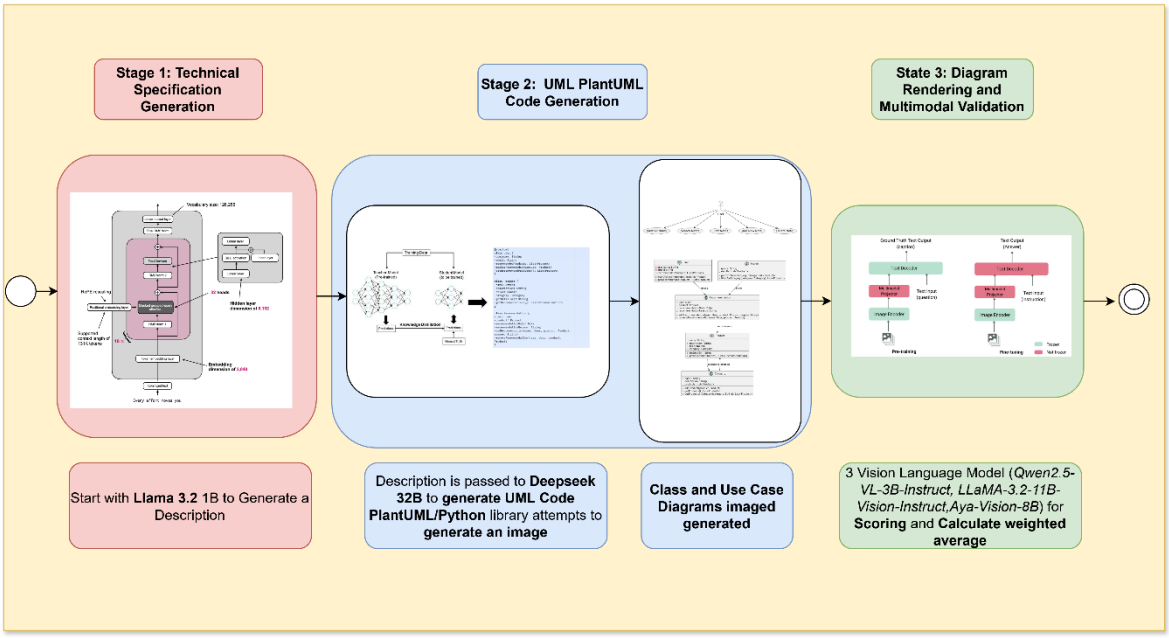

Figure 2. The unified pipeline for UML diagram generation and multimodal scoring

Algorithm 1. Dual-LLM UML generation with multimodal validation

Input: Natural language software requirements R
Output: Validated UML diagrams D and evaluation scores S
Begin
Initialize LLM_spec (lightweight specification model).
Initialize LLM_reason (reasoning-oriented UML synthesis model).
Initialize the set of Vision Language Models: V = {VLM_1, VLM_2, ..., VLM_n}.

```
        Initialize aggregation weights: W = {w₁, w₂, ..., wₙ}.
        Stage 1: Requirement Specification
                StructuredSpec ← LLM_spec(R)
        Stage 2: UML Code Synthesis
                UML_Code ← LLM_reason(StructuredSpec)
                Diagram_Image ← Render(UML_Code)
        Stage 3: Multimodal Validation
                For i = 1 to 8000 do
                        Scoreᵢ ← VLMᵢ(Diagram_Image, StructuredSpec)
                End
                AggregatedScore = Σ_{i=1}^{N} wᵢ × Scoreᵢ
        Return  Diagram_Image, AggregatedScore
End
```

While Figure 2 provides a high-level conceptual overview of the three-stage pipeline, the detailed operational workflow and decision logic are illustrated in Figure 3. This workflow specifically highlights the automated validation process in Stage 3: if the PlantUML code fails to render (the 'NO' path), the system assigns a score of zero and preserves the record for error analysis. Conversely, successfully rendered diagrams are forwarded to the VLM ensemble for multi-dimensional semantic scoring and final aggregation.

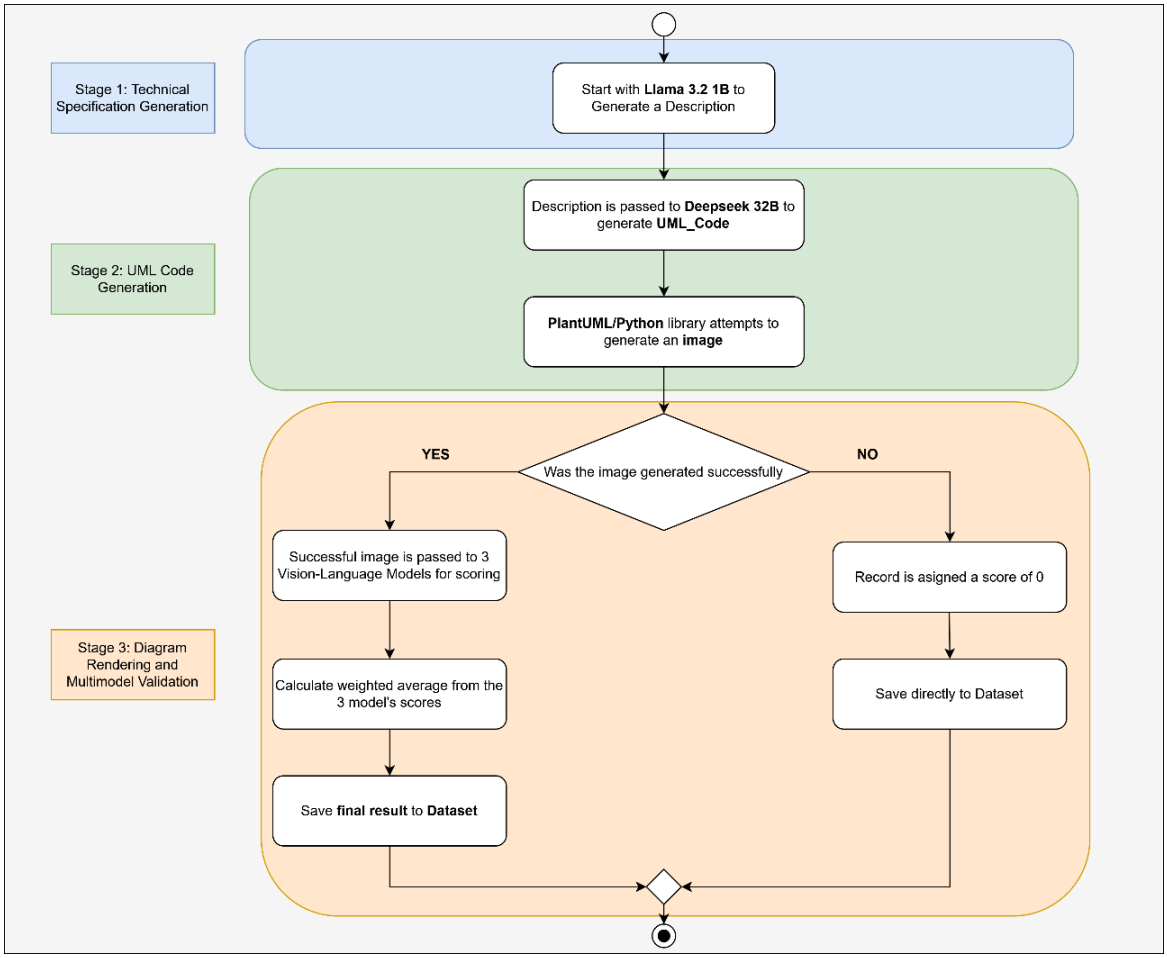

Figure 3. Workflow of the proposed three-stage pipeline

3.6. Implementation details

The experimental setup, including the hardware and software configurations utilized for model training and inference, is summarized in Table 2. We leveraged NVIDIA RTX A5000 GPUs to manage the computational demands and inference latency of the 32B DeepSeek-R1 model. On the software side, the

Unsloth library was integrated to ensure memory-efficient processing, thereby facilitating reproducibility for future research.

Table 2. Experiment environment

Hardware	Software and training
1. 4×NVIDIA RTX A5000 (24GB VRAM): High-performance GPU for deep learning. 2. 2×Threadripper PRO 5965WX (48 cores): multi-core CPU for parallel processing. 3. 256 GB RAM: large memory for handling big datasets.	1. Libraries\& Frameworks: Unsloth 2025.6.2: Fast Mllama patching. Transformers: 4.51.3. vLLM: 0.9.1. Platform: Linux. Torch: 2.7.0+cu126. CUDA: 8.6. CUDA. 2. Training duration: ~160 hours.

4. RESULTS AND DISCUSSION

The experimental findings confirm the feasibility and scalability of automated UML generation through the proposed dual-LLM pipeline combined with multimodal validation. Across both datasets use case (3,000 samples) and class diagrams (5,000 samples) the framework demonstrated strong capability in generating high-quality UML artifacts and systematically verifying their semantic alignment with textual requirements. The combination of LLaMA-3.2-1B-Instruct for lightweight specification generation and DeepSeek-R1-Distill-Qwen-32B for structured UML synthesis proved effective, with 95.8% of class diagram specifications successfully rendered and the majority of use case outputs achieving moderate-to-high fidelity. This scalability illustrates the suitability of the approach for both behavioral and structural modeling tasks, although residual challenges remain in capturing fine grained architectural semantics, such as distinguishing between aggregation and composition.

Comparison between class diagram and use case in Figure 4, the multimodal validation pipeline reinforced this robustness by revealing model specific tendencies in evaluation. For class diagrams, Qwen2.5-VL-3B exhibited a strong optimistic bias, with the majority of outputs rated as perfect (69.1% at score 6), suggesting a tendency to overestimate structural fidelity. LLaMA-3.2-11B-Vision provided conservative but balanced judgments, concentrating in the mid-to-high range (4–5) with no perfect alignments, while Aya-Vision-8B displayed a polarized distribution, with 30% failures (scores 0–1) alongside 28.7% strong evaluations (score 5). In the use case diagram dataset, distributions diverged: LLaMA-3.2-11B-Vision achieved the most consistent performance, with nearly 89% of outputs clustered at score 5, reflecting stable semantic alignment. Qwen2.5-VL-3B peaked at score 4 (64.2%) but spread across lower bands (0–3), while Aya-Vision-8B was unstable, with almost half of outputs failing (45.7% at score 0) yet still assigning high ratings in select cases (20.3% at score 5).

Taken together, these results highlight three key insights. First, Qwen2.5-VL-3B excels in high confidence structural evaluation, making it more suitable for class diagram verification. Second, LLaMA-3.2-11B-Vision provides the most stable and reliable assessments for behavioral modeling, aligning well with the interpretability of use case diagrams. Third, Aya-Vision-8B demonstrates sensitivity but lacks robustness, underscoring the need for refinement to reduce evaluation variance. Importantly, the weighted MMMU aggregation mitigates these biases by integrating complementary strengths optimism from Qwen2.5, conservatism from LLaMA-3.2, and partial adaptability from Aya-Vision-8B producing a more balanced and reliable quality metric.

Overall, distribution of final weighted composite scores in Figure 5, the analysis suggests that behavioral diagrams (use case diagram) are inherently easier for LLMs to synthesize and validate due to their alignment with natural user intent, whereas structural diagrams (class diagram) demand deeper logical reasoning and expose current limitations in semantic granularity. These findings underscore the necessity of multimodal, weighted evaluation pipelines for achieving scalable and trustworthy UML automation, providing strong evidence for the framework's applicability in advancing AI-assisted software engineering.

Placing the results in the context of recent journal and conference research highlights both alignment with and advances beyond prior work on automated UML generation. Early rule-based approaches, such as the heuristic-driven method proposed by Abdelnabi *et al.* [6], achieved reasonable syntactic correctness for UML Class Diagrams but exhibited limited robustness when handling informal or evolving requirements. In contrast, the proposed dual-LLM pipeline reduces dependency on handcrafted rules by leveraging instruction-tuned language models, enabling more flexible generalization across diverse specifications. Machine learning–based methods have sought to improve scalability by classifying requirement sentences into predefined UML elements. Meng and Ban [19] reported promising accuracy using structured NLP techniques; however, such approaches remain constrained by fixed taxonomies and manual feature engineering. The findings of this study indicate that separating requirement interpretation from reasoning-oriented UML synthesis allows LLMs to capture higher-level architectural semantics without explicit rule

design. Recent investigations into LLMs have further demonstrated their potential for direct text-to-diagram generation.

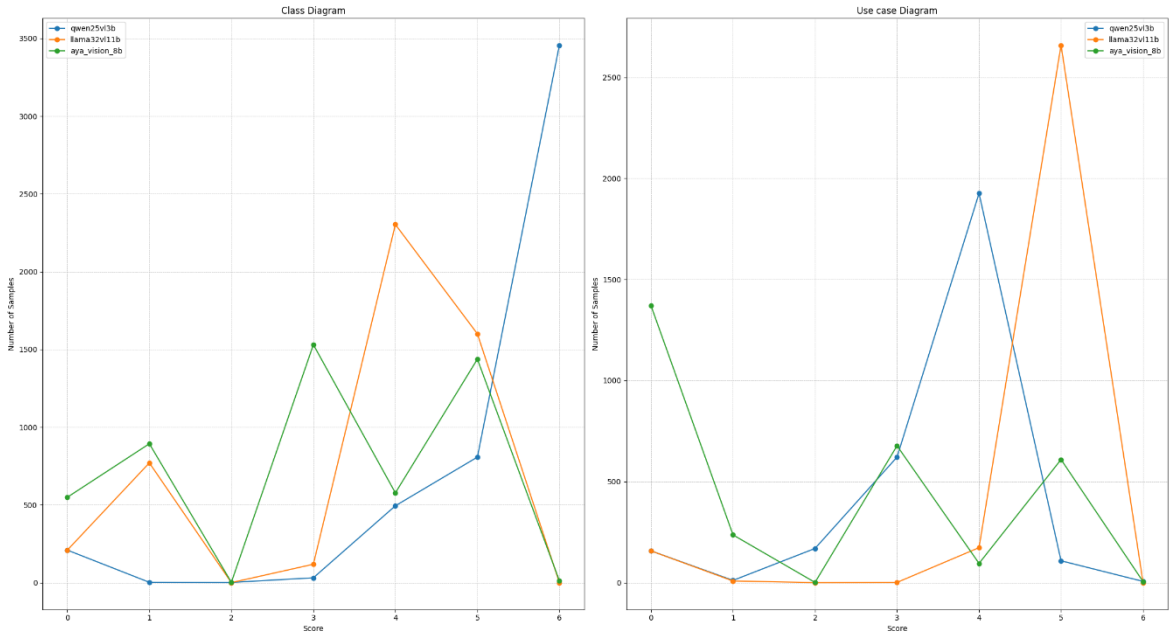

Figure 4. Distribution points of three visual language models for class and use case diagrams

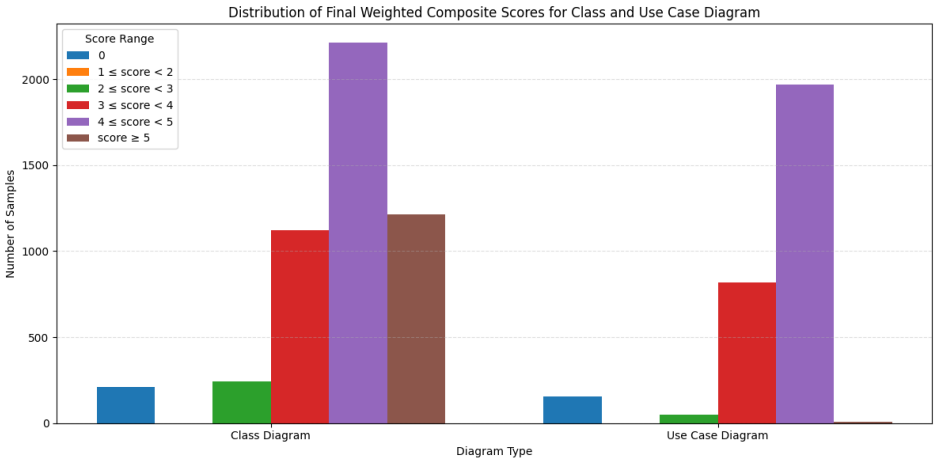

Figure 5. Distribution of final weighted composite scores

Jahan *et al.* [5] showed that generative AI models outperform traditional rule-based techniques for UML sequence diagram derivation, while Bari *et al.* [21] reported that LLM-generated diagrams often conform syntactically to UML standards but still suffer from logical inconsistencies. The results of the present study are consistent with these observations and further suggest that behavioral artifacts, such as use case diagrams, exhibit stronger semantic alignment than structural diagrams. A key distinction of this work lies in its evaluation methodology. Most prior studies rely on text-based similarity metrics, such as BLEU [36] and ROUGE [37], or manual expert assessment, which are insufficient for evaluating the semantic and structural correctness of visual models. By integrating multiple vision–language models through a weighted aggregation mechanism, the proposed multimodal validation framework provides a more objective and robust assessment of UML diagram quality, aligning with recent advances in vision–language evaluation research [17], [18]. Finally, the release of large-scale, multimodally validated benchmark datasets addresses a critical limitation highlighted in recent surveys on UML automation [3], [38]. Compared with existing

datasets that are limited in scale or lack semantic verification, the datasets introduced in this study support reproducible evaluation and enable future research on scalable, reasoning-aware UML automation in AI-driven software engineering.

5. CONCLUSION

This study set out to examine whether a dual-LLM generation pipeline combined with multimodal validation can provide a scalable and reliable solution for automated UML modeling. The findings demonstrate that decomposing the process into lightweight requirement specification and reasoning-oriented UML synthesis significantly improves generation robustness, while multimodal visual validation enables objective assessment beyond traditional text-based metrics. By introducing a reproducible end-to-end framework and releasing large-scale, validated datasets for both class and use case diagrams, this work advances the state of AI-assisted software modeling toward practical adoption. The proposed approach reduces manual modeling effort and mitigates documentation drift, thereby supporting modern Agile and DevOps workflows where requirements evolve rapidly. Despite the high performance of the dual-LLM pipeline, certain limitations persist. First, the framework's ability to distinguish between fine-grained semantic nuances, such as "composition" versus "aggregation" in complex class hierarchies, remains a challenge due to the inherent ambiguity of natural language descriptions. Second, the current model's reasoning stage is highly dependent on the quality of the initial specification generated in Stage 1; any omission in the technical requirements may propagate into the final diagram. Lastly, while the system excels at class and use case diagrams, its scalability to dynamic models with strict temporal constraints, such as Sequence or State diagrams, requires further refinement of the reasoning-oriented synthesis module. Future research will extend this framework to more complex behavioral diagrams, such as sequence and activity diagrams, incorporate multilingual and domain-specific prompting strategies, and explore feedback-driven self-correction mechanisms to further enhance automation accuracy and adaptability in real-world software engineering environments.

FUNDING INFORMATION

This research was funded by the T2026-07-02 initiative at Thai Nguyen University of Information and Communication Technology in Vietnam, with additional support from the AI in Software Engineering Lab.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Van-Viet Nguyen	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Huu-Khanh Nguyen				✓	✓	✓		✓	✓	✓				
Kim-Son Nguyen		✓	✓	✓	✓	✓				✓				
Thi Minh-Hue Luong		✓	✓	✓	✓	✓				✓				
Duc-Quang Vu			✓		✓	✓	✓		✓	✓				
The-Vinh Nguyen	✓	✓			✓	✓		✓	✓	✓	✓	✓		✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

No authors declare conflicts of interest.

INFORMED CONSENT

We have obtained informed consent from all individuals included in this study.

ETHICAL APPROVAL

This research does not require ethical approval as it does not involve human participants, animal subjects, or sensitive data.

DATA AVAILABILITY

Data available in huggingface: https://huggingface.co/nguyenvanviet/datasets.

REFERENCES

[1] G. Booch, J. Rumbaugh, and I. Jacobson, *The unified modeling language user guide*, 2nd ed. Springer Berlin Heidelberg, 1998, doi: 10.1007/3-540-46852-8_1.

[2] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Class Diagram Similarity Measurement: A Different Approach," in *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*, Yogyakarta, Indonesia, 2018, pp. 215-219, doi: 10.1109/ICITISEE.2018.8721021.

[3] S. Ahmed, A. Ahmed, and N. U. Eisty, "Automatic transformation of natural to unified modeling language: a systematic review," in *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)*, IEEE, May 2022, pp. 112–119, doi: 10.1109/SERA54885.2022.9806783.

[4] A. Ferrari, S. Abualhaija, and C. Arora, "Model Generation with LLMs: From Requirements to UML Sequence Diagrams," in *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, Reykjavik, Iceland, 2024, pp. 291-300, doi: 10.1109/REW61692.2024.00044.

[5] M. Jahan *et al.*, "Automated derivation of uml sequence diagrams from user stories: unleashing the power of generative ai vs. a rule-based approach," in *Proceedings - MODELS 2024: ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, in MODELS '24. ACM, 2024, pp. 138–148, doi: 10.1145/3640310.3674081.

[6] E. A. Abdelnabi, A. M. Maatuk, T. M. Abdelaziz, and S. M. Elakeili, "Generating UML class diagram using NLP techniques and heuristic rules," in *Proceedings - STA 2020: 2020 20th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering*, IEEE, Dec. 2020, pp. 277–282, doi: 10.1109/STA50679.2020.9329301.

[7] D. K. Deeptimahanti and R. Sanyal, "Semi-automatic generation of uml models from natural language requirements," in *Proceedings of the 4th India Software Engineering Conference 2011, ISEC'11*, in ISEC '11. ACM, Feb. 2011, pp. 165–174, doi: 10.1145/1953355.1953378.

[8] X. Hou *et al.*, "Large language models for software engineering: a systematic literature review," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 8, pp. 1–79, Nov. 2024, doi: 10.1145/3695988.

[9] M. Alenezi and M. Akour, "AI-driven innovations in software engineering: a review of current practices and future directions," *Applied Sciences*, vol. 15, no. 3, pp. 1-26, Jan. 2025, doi: 10.3390/app15031344.

[10] B. Wang, C. Wang, P. Liang, B. Li, and C. Zeng, "How llms aid in uml modeling: an exploratory study with novice analysts," in *Proceedings - 2024 IEEE International Conference on Software Services Engineering, SSE 2024*, IEEE, 2024, pp. 249–257, doi: 10.1109/SSE62657.2024.00046.

[11] Z. Babaalla, H. Abdelmalek, A. Jakimi, and M. Oualla, "Extraction of uml class diagrams using deep learning: comparative study and critical analysis," *Procedia Computer Science*, vol. 236, pp. 452–459, 2024, doi: 10.1016/j.procs.2024.05.053.

[12] L. Yuan *et al.*, "Improving natural language understanding for llms via large-scale instruction synthesis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 24, pp. 25787–25795, Apr. 2025, doi: 10.1609/aaai.v39i24.34771.

[13] P. Wang *et al.*, "Qwen2-vl: enhancing vision-language model's perception of the world at any resolution," *arXiv*, Dec. 2024, doi: 10.48550/arXiv.2409.12191.

[14] A. Üstün *et al.*, "Aya model: an instruction finetuned open-access multilingual language model," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024, pp. 15894–15939, doi: 10.18653/v1/2024.acl-long.845.

[15] A. Bates, R. Vavricka, S. Carleton, R. Shao, and C. Pan, "Unified modeling language code generation from diagram images using multimodal large language models," *Machine Learning with Applications*, vol. 20, pp. 1-17, Jun. 2025, doi: 10.1016/j.mlwa.2025.100660.

[16] J. Torcal, V. Moreno, J. Llorens, and A. Granados, "Creating and validating a ground truth dataset of unified modeling language diagrams using deep learning techniques," *Applied Sciences*, vol. 14, no. 23, pp. 1-12, Nov. 2024, doi: 10.3390/app142310873.

[17] S. Danish, A. Sadeghi-Niaraki, S. U. Khan, L. M. Dang, L. Tightiz, and H. Moon, "A comprehensive survey of vision–language models: pretrained models, fine-tuning, prompt engineering, adapters, and benchmark datasets," *Information Fusion*, vol. 126, Feb. 2026, doi: 10.1016/j.inffus.2025.103623.

[18] C. Picard *et al.*, "From concept to manufacturing: evaluating vision-language models for engineering design," *Artificial Intelligence Review*, vol. 58, no. 9, pp. 1-75, Jul. 2025, doi: 10.1007/s10462-025-11290-y.

[19] Y. Meng and A. Ban, "Automated uml class diagram generation from textual requirements using nlp techniques," *International Journal on Informatics Visualization*, vol. 8, no. 3–2, pp. 1905–1915, Nov. 2024, doi: 10.62527/joiv.8.3-2.3482.

[20] M. Evtikhiev, E. Bogomolov, Y. Sokolov, and T. Bryksin, "Out of the bleu: how should we assess quality of the code generation models?," *Journal of Systems and Software*, vol. 203, Sep. 2023, doi: 10.1016/j.jss.2023.111741.

[21] D. D. Bari, G. Garaccione, R. Coppola, M. Torchiano, and L. Ardito, "Evaluating large language models in exercises of uml class diagram modeling," in *International Symposium on Empirical Software Engineering and Measurement*, in ESEM '24. ACM, Oct. 2024, pp. 393–399, doi: 10.1145/3674805.3690741.

[22] N. Bouali, M. Gerhold, T. Rehman, and F. Ahmed, "Toward automated UML diagram assessment: comparing LLM-generated scores with teaching assistants," in *Proceedings of the 17th International Conference on Computer Supported Education*, *SCITEPRESS - Science and Technology Publications*, 2025, pp. 158–169, doi: 10.5220/0013481900003932.

[23] S. Yang and H. Sahraoui, "Towards automatically extracting UML class diagrams from natural language specifications," in *Proceedings - ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems, MODELS 2022: Companion Proceedings*, in MODELS '22. ACM, Oct. 2022, pp. 396–403, doi: 10.1145/3550356.3561592.

[24] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *International Journal of Robotics Research*, vol. 37, no. 4–5, pp. 513–542, Oct. 2018, doi: 10.1177/0278364917734298.

[25] A. M. Maatuk and E. A. Abdelnabi, "Generating UML use case and activity diagrams using NLP techniques and heuristics rules,"

in *ACM International Conference Proceeding Series*, in DATA'21. ACM, Apr. 2021, pp. 271–277, doi: 10.1145/3460620.3460768.
[26] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.
[27] H. Touvron *et al.*, "LLaMA 2: open foundation and fine-tuned chat models," *arXiv*, 2023, doi: 10.48550/arXiv.2307.09288.
[28] H. Touvron, "LLaMA: open and efficient foundation language models," *arXiv*, 2023.
[29] A. Conrardy and J. Cabot, "From image to UML: first results of image-based UML diagram generation using LLMs," *CEUR Workshop Proceedings*, vol. 3727, pp. 55–65, 2024.
[30] M. I. Mukhtar, B. S. Galadanci, and E. Addresses, "Automatic code generation from UML diagrams: the state-of-the-art," *Science World Journal*, vol. 13, no. 4, pp. 47–60, 2018.
[31] B. Karasneh and M. R. V. Chaudron, "Img2UML: a system for extracting UML models from images," in *Proceedings - 39th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2013*, IEEE, 2013, pp. 134–137, doi: 10.1109/SEAA.2013.45.
[32] Y.-C. Chen *et al.*, "UNITER: universal image-text representation learning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12375 LNCS, Springer International Publishing, 2020, pp. 104–120, doi: 10.1007/978-3-030-58577-8_7.
[33] P. Zhang *et al.*, "VinVL: revisiting visual representations in vision-language models," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2021, pp. 5575–5584, doi: 10.1109/CVPR46437.2021.00553.
[34] F. Chen, L. Zhang, X. Lian, and N. Niu, "Automatically recognizing the semantic elements from UML class diagram images," *Journal of Systems and Software*, vol. 193, Nov. 2022, doi: 10.1016/j.jss.2022.111431.
[35] J. Bai *et al.*, "Qwen-vl: a versatile vision-language model for understanding, localization, text reading, and beyond," *arXiv*, 2023, doi: 10.48550/arXiv.2308.12966.
[36] K. Papineni, S. Roukos, T. Ward, and W. J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, in ACL '02, vol. 2002- July. Association for Computational Linguistics, 2002, pp. 311–318, doi: 10.3115/1073083.1073135.
[37] C.-Y. Lin, "ROUGE: a package for automatic evaluation of summaries," in *Text Summarization Branches Out*, 2001, pp. 74–81.
[38] E. A. Abdelnabi, A. M. Maatuk, and M. Hagal, "Generating uml class diagrams from natural language requirements: a survey of approaches and techniques," in *2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering, MI-STA 2021 - Proceedings*, IEEE, May 2021, pp. 288–293, doi: 10.1109/MI-STA52233.2021.9464433.
[39] D. Guo *et al.*, "DeepSeek-r1: incentivizing reasoning capability in llms via reinforcement learning," *ArXiv*, vol. 500, pp. 1–22, 2025, doi: 10.48550/arXiv.2501.12948.
[40] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *Advances in Neural Information Processing Systems 35*, in NeurIPS 2022. San Diego, California, USA: Neural Information Processing Systems Foundation, Inc. (NeurIPS), 2022, pp. 24824–24837, doi: 10.52202/068431-1800.
[41] Y. Liu *et al.*, "MMBench: is your multi-modal model an all-around player?," *Lecture Notes in Computer Science*, vol. 15064 LNCS, pp. 216–233, 2025, doi: 10.1007/978-3-031-72658-3_13.
[42] V.-V. Nguyen *et al.*, "A novel unified framework for automated generation and multimodal validation of uml diagrams," *Computer Modeling in Engineering & Sciences*, vol. 146, no. 1, pp. 1–10, 2026, doi: 10.32604/cmes.2025.075442.
[43] X. Yue *et al.*, "MMMU: a massive multi-discipline multimodal understanding and reasoning benchmark for expert agi," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2024, pp. 9556–9567, doi: 10.1109/CVPR52733.2024.00913.

BIOGRAPHIES OF AUTHORS

Van-Viet Nguyen is a researcher and Ph.D. student at the Thai Nguyen University of Information and Communication Technology, Thai Nguyen, Vietnam. He received a bachelor's in Information Technology at Thai Nguyen University (ICTU), Vietnam in 2009. He got a master's degree on Information Technology at Manuel S. Enverga University, Philippines in 2012. His research interests include artificial intelligence, machine learning, and generative AI. He can be contacted at email: nvviet@ictu.edu.vn.

Huu-Khanh Nguyen has graduated with a Master's degree in Computer Science from the University of Information and Communications Technology-Thai Nguyen University since 2022 and is currently a Ph.D. student here since 2023. His main research interests are computer science, natural language processing, generative AI, and computer vision. He can be contacted at email: khanhnh@tnu.edu.vn.

Kim-Son Nguyen received her Bachelor of Information Technology from Thai Nguyen University of Information Technology in 2009 and her Master of Information Technology from Manuel S. Enverga University, Philippines, in 2012. Currently, she is a lecturer at the Thai Nguyen University of Information and Communication Technology, Vietnam. Her main research interests are artificial intelligence, large language models, and geographic information systems. In the current study, she contributed to conceptualization, methodology, software development, and original draft preparation. She is currently leading a research project on the integration of open-source language models with GIS applications for resource-constrained environments. She can be contacted at email: nkson@ictu.edu.vn.

Thi Minh-Hue Luong received her Bachelor of Information Technology from Thai Nguyen University of Information Technology in 2010 and her Master of Information Technology from Manuel S. Enverga University, Philippines, in 2013. She has been a lecturer at the Faculty of Information Technology, Thai Nguyen University of Information Technology since 2010. Her research interests include computer science, artificial intelligence, and communication technology. For this research, she contributed to conceptualization, investigation, validation, and supervision of the project. She has experience in managing interdisciplinary research projects combining AI and geospatial technologies. She can be contacted at email: lmhue@ictu.edu.vn.

Duc-Quang Vu was born in Nam Dinh, Vietnam in 1991. He received a B.S. degree in education in information technology from the Thai Nguyen University of Education, Vietnam, in 2013 and an M.S. degree in Information Systems, from the University of Engineering and Technology (UET), Vietnam National University, Hanoi (VNU) in 2016. He received the Ph.D. degree in the Department of Computer Science and Information Engineering, National Central University, Taiwan in 2022 and a postdoc in 2023. His research interests include machine learning, deep learning, computer vision, speech processing, and bioinformatics. He can be contacted at email: vdquang@ictu.edu.vn.

The-Vinh Nguyen is currently a senior lecturer at the Faculty of Information Technology, University of Information and Communication Technology. He graduated with a master's degree in information systems management from Oklahoma State University, USA (under scholarship 322). He completed his Ph.D. program under Project 911 in 2020 at Texas Tech University, USA. His main research interests are computer vision, computer visualization, and computer in human behavior. He has authored or coauthored more than 50 publications with 16 H-index and more than 850 citations. He can be contacted at email: vinhnt@ictu.edu.vn.