Evaluating and Enhancing Large Language Models in Generating UML Class Diagram for Good Code Design

Delivering a high-quality software product requires major emphasis on software analysis and design. The design phase forms a strong base for achieving quality in downstream artifacts, where tools like UML enable clear and concise documentation. Particularly, a UML class diagram act as a valuable asset in bridging the gap between requirements and implementation, thereby guiding the developers during code by capturing the key structural elements such as classes, attributes, methods, interfaces, and relationships.

However, in practice, creating architecturally and semantically sound class structures that align with OOAD principles requires a diverse set of skills. This is because of the complexity of iterative and interactive nature of design process. Therefore, automation of this process could save significant time and resources. Recently, large language models have shown great potential in automating software engineering tasks including generation of UML class diagrams for software architecture and design. However, the effectiveness of understanding their capabilities in this domain remain limited due to the complexity of iterative, artifact-rich workflows. Thus, a simplification is needed to isolate and better understand the foundational capabilities and limitations of LLMs in generating architecturally meaningful class structures.

To make progress, we adopt a controlled, initial experimental setup in this study. We deliberately simplify the process by restricting the input modality to design-oriented text. We acknowledge that this does not fully reflect realistic design process, however, it provides a constrained testbed to argue: if LLMs can't handle structured design-oriented text properly, we can't expect them to cope with messy, iterative, artifact-rich workflows.

Our approach provides an experimental baseline consisting of two complementary strategies. First, we investigate whether explicitly incorporating design rules, drawn from published OOAD literature, into the prompt improves the quality of LLM-generated diagrams. Notably, we don't claim it as a novel contribution. Instead, it serves as a fairness baseline, because most of the prior work (i.e. standard baseline) has expected the LLMs to produce OOAD-aligned diagrams without explicitly passing design rules to them. Second, we propose a novel preference-based few-shot prompting approach that aligns LLM outputs with established design principles using a synthetic dataset constructed from annotated modeling examples. In this dataset, LLMs are exposed to the pairs of modeling solutions, one aligned with design principles and one not, to help them learn preference for architecturally sound outputs.

We evaluate both strategies and compare them against standard baseline tools. Our results show that preference-aligned prompting significantly improves structural quality, particularly in class abstraction, interface use, and relationship modeling; reduce design smells and post-correction efforts, thereby advancing towards trustworthy AI assistants in software design.

Metric	Standard Baseline	Fairness Baseline	Proposed Approach
Structural Correctness Score	Classes/Attributes/Methods: moderate; Relationships: weak	Slightly improved attributes/methods	Strong gains in structural elements; Slightly weaker attributes/methods
Principle Adherence Score	Avg = 0.1687	Avg = 0.2016	0.769 (over 3× higher than best baseline)
Expert Usefulness Ratings	Top = 3.0; Avg = 2.66	Top = 3.0; Avg = 2.75	3.33 (highest, only minor revisions needed)
Principle Adherence Improvement	—	+0.0329 vs. Standard	+0.6003 vs. Standard; +0.5674 vs. Fairness
Post-Correction Effort	22 edits required	17 edits required	7 edits required (68.2% gain vs. Standard, 58.8% vs. Fairness)