Prompt Engineering Applied to the Generation of Software Modeling Documents: A Systematic Mapping Study

Rafael Ribas de Lima
Graduate Program in Computer Science, Federal University of Technology – Paraná (UTFPR)
Ponta Grossa, PR, Brazil
rafaellima.2025@alunos.utfpr.edu.br

Simone Nasser Matos
Graduate Program in Computer Science, Federal University of Technology – Paraná (UTFPR)
Ponta Grossa, PR, Brazil
snasser@utfpr.edu.br

Abstract

Automating software modeling from natural language specifications is a central challenge in Model-Driven Engineering. Large Language Models (LLMs) and Prompt Engineering techniques have emerged as resources to automate this task, but the literature on the topic remains fragmented across different diagram types, representation languages, and instruction strategies. This paper presents a Systematic Mapping Study on the use of LLMs and Prompt Engineering for the generation, modification, and validation of software modeling documents. From 475 studies identified in three digital libraries (Scopus, IEEE Xplore, and ScienceDirect), 52 primary studies published between 2023 and 2026 were included after applying inclusion and exclusion criteria and full-text reading. Results indicate that GPT-family models remain predominant as the generation engine (73.1% of the studies), that Class diagrams and Architecture and Component models concentrate most of the produced artifacts, and that Few-shot, Chain-of-Thought, and prompt chaining strategies coupled with deterministic syntactic validators constitute the most reported pattern to contain structural conformance errors. The mapping also identifies a persistent gap in semantic validation and explainability (XAI) mechanisms for the generated artifacts.

Keywords

Prompt Engineering, Large Language Models, Software Modeling, UML, Systematic Mapping Study

1 Introduction

The development of complex software systems depends on modeling activities and formal requirements specification to ensure quality, maintainability, and alignment with business needs. The Unified Modeling Language (UML) and other modeling languages structure both static aspects (class and component diagrams) and behavioral aspects (sequence, activity, and use case diagrams) of a system. Traditionally, the transition from natural language requirements to valid formal models demands high cognitive effort and specialized technical knowledge, making the process prone to errors and inconsistencies [10, 19].

The emergence of Large Language Models (LLMs) and Prompt Engineering introduced a new paradigm to automate this transition, replacing part of the manual construction of diagrams with workflows assisted by generative artificial intelligence [29]. However, generating modeling artifacts imposes requirements distinct from free-form text or source code generation: it demands strict syntactic conformance to target metamodels and relational integrity among entities. Attempts to use direct, monolithic single-turn prompts frequently result in structural conformance failures [2, 52], and the models exhibit semantic vulnerabilities reported in the literature, such as hallucination of concepts and entities that do not exist in the target metamodel [17, 39].

Related secondary studies address generative AI in adjacent contexts but do not cover the generation of modeling documents through prompt engineering: Esposito et al. [12] synthesize GenAI across the software architecture life cycle; Liu et al. [27] catalog architectural patterns for foundation-model-based agents; and Su et al. [48] map technology trends in software architecture. None of them investigates the instruction strategies employed to generate, modify, or validate UML diagrams, OCL constraints, or business process models, a gap this mapping aims to fill. This work presents a Systematic Mapping Study (SMS) that consolidates this knowledge by answering four research questions about the technologies employed, the instruction strategies used, the treatment of input requirements, and the resulting modeling artifacts. The remainder of this paper is organized as follows: Section 2 presents the background; Section 3 describes the methodological protocol; Section 4 presents the results; Sections 5 and 6 discuss the findings and the threats to validity; and Section 7 concludes the paper.

2 Background

Large Language Models are built on neural network architectures based on attention mechanisms (Transformers), trained on large volumes of textual and source code data. In Software Engineering, these models have evolved from text auto-completion assistants into generative engines capable of translating informal requirements into executable code, design specifications, and technical documentation [14, 56].

Prompt Engineering consists of the systematic design of instructions, rules, and constraints provided as input to the models, steering their behavior without retraining weights. The strategies most frequently reported in the analyzed studies (Section 4.3) include *Zero-Shot* (direct instruction only [13]), *Few-Shot* (input–output examples illustrating the expected format [26]), *Chain-of-Thought* (CoT, intermediate reasoning steps before the final answer [46]), and *Prompt Chaining* (decomposition of a complex goal into sequential steps, each consuming the previous output [28, 57]).

Model-Driven Engineering (MDE) elevates conceptual models to first-class artifacts of the engineering process, from which code, documentation, and data schemas can be derived through systematic transformations [11]. UML and OCL describe, respectively, a system's structure/behavior and the formal constraints ensuring its logical consistency [1, 47]. The main challenge in integrating MDE and LLMs lies in strict conformance to rigid metamodels: while

traditional MDE relies on deterministic parsers, LLMs produce outputs stochastically, which demands hybrid architectures capable of automatically validating and repairing the generated artifacts [52].

3 Methodology

This mapping was conducted following the guidelines of Petersen et al. [42] and Kitchenham and Charters [22] for secondary studies in Software Engineering, structured in three stages: (i) protocol planning; (ii) search execution and screening; and (iii) data extraction and synthesis.

3.1 Research Questions

Four research questions (RQs) guided the mapping: **RQ1** – Which generative AI models and supporting tools are employed in software modeling automation? **RQ2** – Which Prompt Engineering techniques guide the interaction with generative models? **RQ3** – How does the literature address the extraction and processing of input requirements? **RQ4** – Which software models and representation languages result from the assisted generation process?

3.2 Search Strategy

The search scope followed the PCC structure (*Population, Concept, Context*), adopted instead of PICOC as more appropriate for a mapping study without a comparison arm: Population (software system modeling processes and requirements specification); Concept (the use of generative AI and Prompt Engineering to generate, modify, or validate modeling artifacts); Context (academic and industrial settings in which AI-assisted software modeling is investigated, restricted to peer-reviewed publications from 2021 to 2026). The search was executed in Scopus, IEEE Xplore, and ScienceDirect, restricted to title, abstract, and keyword metadata, with the following string:

> (*"Generative AI" OR "Large Language Models" OR "LLM" OR "GPT" OR "Chat-GPT"*) AND (*"Prompt Engineering" OR "Prompting" OR "In-context learning"*) AND (*"Software Modeling" OR "System Modeling" OR "UML" OR "Software Architecture" OR "System Design"*)

Scopus, IEEE Xplore, and ScienceDirect were selected for their broad and complementary coverage of Software Engineering and Artificial Intelligence venues. Scopus and ScienceDirect share the same editorial group (Elsevier), which entails an expected degree of overlap between the two sources; Scopus was nonetheless retained because it additionally indexes publishers beyond Elsevier (e.g., IEEE, Springer), while ScienceDirect provides deeper full-text coverage of Elsevier venues. Because ScienceDirect enforces a maximum of eight boolean operators per query, the string was adapted to (*"Generative AI" OR LLM OR GPT*) AND (*"Prompt Engineering" OR Prompting*) AND (*"Software Modeling" OR UML OR "Software Architecture"*), retaining the broadest root term per PCC facet.

3.3 Selection Criteria

Table 1 summarizes the five Inclusion Criteria (IC) and seven Exclusion Criteria (EC) applied during screening.

Table 1: Inclusion and exclusion criteria

ID	Description
IC1	Generative AI/LLMs for requirements extraction, creation, modification, or analysis of software models
IC2	Prompt Engineering techniques aimed at modeling
IC3	Peer-reviewed publications
IC4	Published between 2021 and 2026
IC5	Empirical evidence, prototypes, or case studies
EC1	Source code generation only, without high-level modeling
EC2	SE tasks unrelated to modeling
EC3	Gray literature
EC4	Languages other than English or Portuguese
EC5	Secondary studies
EC6	Duplicates
EC7	Article not publicly accessible

3.4 Screening and Extraction

The search and screening were executed between March 1 and May 31, 2026 using the Parsif.al platform[1], following the PRISMA 2020 (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*) flow [40], summarized in Figure 1. Duplicate records were identified and removed automatically by the platform, without manual re-verification. The remaining records were screened by title and abstract against the IC/EC criteria (Table 1), and all retained studies underwent full-text reading, followed by a quality assessment based on four criteria: description of the prompt strategy, specification of the model and its settings, presence of rigorous validation of the generated artifacts, and availability of research artifacts. Each criterion was scored as Yes (1.0), Partially (0.5), or No (0.0) (maximum 4.0; cutoff 2.0). All nine full-text-stage exclusions resulted from EC7 (article not publicly accessible); none was due to the quality-assessment cutoff. Screening, extraction, and categorization were conducted by a single researcher (see Section 6, Internal Validity, for the mitigation measures adopted).

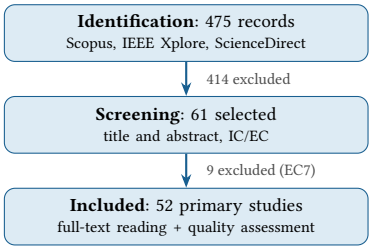

Figure 1: Study selection flow

Data extraction followed a structured form (year, AI model, supporting tool and architecture, diagram type, representation language, prompting strategy, validation method, and reported limitations), applied and manually audited against the full text of the 52 included studies; the frequency statistics in Section 4 therefore refer to the entire corpus. The analysis followed the faceted classification scheme of Petersen et al. [42]. The 52 primary studies fully compose the reference list of this paper and are identified by the labels P1–P52 (Table 2), adopted throughout Sections 4 and 5 to attribute each finding to the studies that evidence it.

[1] https://parsif.al

Table 2: Primary studies included in the mapping, identified as P1–P52

ID	Study	ID	Study	ID	Study	ID	Study
P1	Wan [52]	P2	Liu [26]	P3	Ferrari [13]	P4	Mayr-Dorn [30]
P5	Maranhão [29]	P6	Rukmono [46]	P7	Eisenreich [10]	P8	Hatahet [18]
P9	Al-Ahmad [3]	P10	Trifan [50]	P11	Nast [37]	P12	Abukhalaf [2]
P13	Rizzi [44]	P14	Zhao [56]	P15	Delgado [8]	P16	Patel [41]
P17	Bates [4]	P18	Dolha [9]	P19	Siala [47]	P20	Mercado [32]
P21	Zhuang [58]	P22	Kim [20]	P23	Nguyen [38]	P24	Kong [23]
P25	Odu [39]	P26	Malamas [28]	P27	Wang [53]	P28	Vitale [51]
P29	Zhang [55]	P30	Muttillo [33]	P31	Tanhaei [49]	P32	Lamas [24]
P33	Calamo [5]	P34	Fokaefs [14]	P35	Naimi [35]	P36	Jin [19]
P37	Rejithkumar [43]	P38	De Vito [7]	P39	Muttillo [34]	P40	Nair [36]
P41	Fresemann [15]	P42	Garaccione [16]	P43	Zou [59]	P44	Erum [11]
P45	Kim [21]	P46	Licardo [25]	P47	Rukmono [45]	P48	Copei [6]
P49	Mazur [31]	P50	Abukhalaf [1]	P51	Gheorghita [17]	P52	Zhu [57]

4 Results

This section presents the findings of the mapping, organized by research question.

4.1 Overview

The 52 included studies were published between 2023 and 2026[2], and concentrate mostly in the last two years of the analyzed period: 50.0% published in 2025 and 23.1% in 2026, against 21.2% in 2024 and 5.8% in 2023. Three aspects contextualize this distribution. First, although the search window opened in 2021 (IC4), no study published before 2023 satisfied the selection criteria, which is consistent with the broad availability of instruction-following LLMs only after late 2022. Second, the growth coincides with the corpus-internal adoption of reasoning-oriented models (P33, P23) [5, 38], whose capabilities broadened the feasibility of higher-abstraction modeling tasks. Third, the lower 2026 count reflects a partial year (coverage through May 31, 2026), not a decline in research interest.

4.2 RQ1 – AI Models and Supporting Tools

Across the 52 included studies, the GPT family appears as the generation engine in 73.1% of the cases, followed by LLaMA (25.0%), Qwen, DeepSeek, and Gemini (17.3% each), Mistral (15.4%), Claude (11.5%), GLM (7.7%), Gemma (5.8%), and Phi (3.8%), as shown in Figure 2 (multiple choice; the sum exceeds 100% because several studies compare more than one model). The predominance of GPT is consistent with its recurring role as a comparison *baseline*, while open-weight models (LLaMA, Qwen, DeepSeek, GLM) appear mostly in multi-agent architectures executed locally via *frameworks* such as Ollama (P45, P2) [21, 26]. This predominance may also partly reflect an availability bias rather than a purely comparative-performance advantage: GPT-family models were among the first LLMs to reach broad public accessibility following the release of ChatGPT in late 2022, increasing their likelihood of adoption as a default choice in early studies independent of comparative performance.

Regarding supporting tools, PlantUML stands out as the predominant interpreter for rendering diagrams from declarative textual syntax (P14, P2) [26, 56], graph databases such as Neo4j are used to consolidate extracted semantic relations (P14) [56], and multi-agent orchestration frameworks coordinate specialized roles in generation pipelines (P45, P52) [21, 57].

[2]25 studies from ScienceDirect, 17 from Scopus, and 10 from IEEE Xplore.

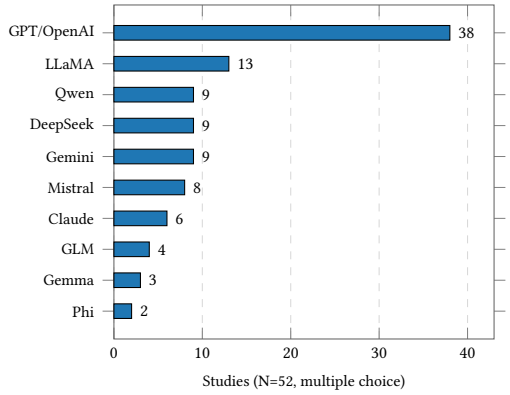

Figure 2: Frequency of AI model families (N=52)

4.3 RQ2 – Prompt Engineering Strategies

Figure 3 shows that Few-shot / *In-Context Learning* appears in 46.2% of the studies, followed by Zero-shot and Persona/Role patterns (38.5% each), Chain-of-Thought (30.8%), Prompt Chaining or feedback loops (17.3%), and Multi-Agent Debate (1.9%). The combined use of these techniques, more than any single one, characterizes the approaches with the highest syntactic reliability: chaining decomposes the generation of a complex diagram into smaller steps, each evaluated before proceeding, while error feedback loops re-inject compiler or parser messages into the prompt for self-repair (P14, P52) [56, 57].

4.4 RQ3 – Requirements Processing and Extraction

The treatment of input requirements follows two predominant approaches. The first decomposes the natural language specification into structured units (entities, relations, flows) before submitting it to the LLM, reducing the cognitive load per inference turn (P14) [56]. The second incorporates ambiguity mitigation mechanisms, such as debate and self-reflection among specialized agents (P2) [26], to handle incomplete or conflicting requirements before generating the final model. Studies in regulatory domains (P21, P31) [49, 58] report the need for additional human validation to handle ambiguous normative requirements.

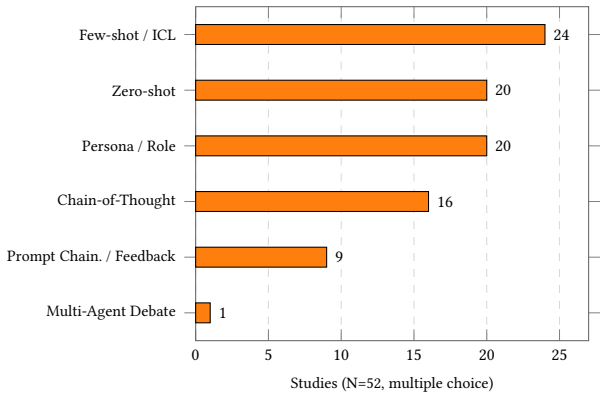

Figure 3: Frequency of prompting strategies (N=52)

4.5 RQ4 – Artifacts and Representation Languages

Table 3 shows that Class diagrams and Architecture and Component models concentrate the highest frequency (25.0% each), followed by Sequence (17.3%) and Use Case diagrams (15.4%). Formal OCL constraints and business processes (BPMN) appear in 9.6% of the studies each; Activity and State diagrams and goal models (GRL) appear in 7.7% each; and niche artifacts – graph representations, Ecore/XMI, and AADL – appear more sparsely, reflecting specific application domains, such as safety-critical embedded systems (P43) [59], business processes (P18, P46) [9, 25], and enterprise modeling (P11) [37].

Table 3: Frequency of modeling artifact types (N=52)

Artifact Type	Freq.	%
Class Diagram	13	25.0%
Architecture / Components	13	25.0%
Sequence Diagram	9	17.3%
Use Case Diagram	8	15.4%
OCL Constraints	5	9.6%
Process Models (BPMN)	5	9.6%
Activity Diagram	4	7.7%
State Diagram	4	7.7%
Goals / GRL	4	7.7%
Graph Representations	3	5.8%
Ecore / XMI	2	3.8%
AADL	1	1.9%

Regarding output representation languages, PlantUML consolidates itself as the predominant format, owing to its concise syntax and low token consumption compared to XML representations; XMI and structured JSON appear in studies prioritizing direct tool integration (P49) [31]; and domain-specific languages compiled by formal grammars appear in studies aimed at regulatory specifications and OCL constraints (P20, P26) [28, 32].

5 Discussion

The findings of this mapping characterize recurring architectural patterns and technical gaps in LLM-assisted software modeling automation.

5.1 Recurring Architectural Patterns

A technically relevant pattern, present in 13 of the 52 studies (25.0%), combines a generative model – predominantly from the GPT family – as a flexible semantic translator, coupled with a deterministic validator that checks syntactic conformance before delivering the artifact to the user (P50, P44) [1, 11]. In the remaining 39 studies (75.0%), validation relies predominantly on similarity metrics or expert judgment, without an explicit syntactic verifier in the generation loop.

This neuro-symbolic coupling, when present, is frequently reinforced by *Prompt Chaining*: instead of delegating the generation of a complex diagram to a single turn, the process is decomposed into smaller sequential steps, each auditable in isolation. Multi-agent frameworks with collaborating specialized roles appear as an extension of this pattern for scenarios with higher requirement uncertainty (P2, P20) [26, 32].

5.2 Structural and Behavioral Modeling

Although structural diagrams (Class, Architecture, Components) and behavioral ones (Sequence, Use Case, Activity, State) appear with comparable frequency in the corpus (Table 3), approximately 27% of the 52 studies (14 occurrences) explicitly report limitations associated with dynamic aspects: dependency directionality, cardinalities, temporal flows, and state transitions (P8, P13, P43, P51) [17, 18, 44, 59]. This suggests the difficulty lies less in how often behavioral diagrams are studied than in the precision of mapping their dynamic relations, an aspect frequency metrics alone do not capture.

5.3 Technical Gaps

Three recurring gaps emerge from the synthesis. First, the validation of generated artifacts concentrates mostly on syntactic conformance (the diagram compiles or is well-formed), with few studies evaluating semantic correctness against the actual intent of the input requirement (P44, P51, P26) [11, 17, 28]. Second, the absence of Explainable AI (XAI) mechanisms limits the auditability of modeling decisions made by LLMs, hindering adoption in regulated domains (P36) [19]. Third, benchmarks and datasets specific to software modeling remain scarce compared to those for source code generation, hindering objective comparisons; one study reports a risk of training-data leakage when reusing benchmarks predating model training cutoffs (P34) [14].

6 Threats to Validity

Following the classification of Wohlin et al. [54], four threat categories are identified and discussed below.

Construct validity. This threat was mitigated by the calibration of the search string and the use of three consolidated indexing libraries. Nevertheless, the restriction to English and Portuguese (EC4) may have excluded relevant studies published in other languages.

Internal validity. Screening, extraction, and categorization were conducted by a single researcher. To mitigate this risk, objective inclusion and exclusion criteria were applied, along with a manual audit of the structured extraction against the full text of the 52 included studies.

Conclusion validity. The categorization of technologies, prompting strategies, and artifact types into aggregated families (Figures 2 and 3, Table 3) involves grouping decisions that may simplify technical variations reported in the original studies.

External validity. The 2023–2026 time frame and the pace of new model releases limit the durability of the technological conclusions, making periodic updates of this mapping advisable.

7 Conclusion and Future Work

This paper presented a Systematic Mapping Study on Prompt Engineering and Large Language Models for generating software modeling documents, based on 475 identified and 52 included primary studies, with data extraction audited for the entire corpus. The four research questions were answered as follows. Regarding RQ1, the GPT family remains the predominant generation engine (73.1% of the studies), while open-weight models gain ground in locally executed multi-agent architectures. Regarding RQ2, Few-shot/ICL (46.2%) and Zero-shot (38.5%) prevail as instruction strategies, but the approaches with the highest reported syntactic reliability combine them with deterministic validators in the generation loop – a neuro-symbolic pattern present in 25.0% of the corpus. Regarding RQ3, requirements processing converges on structured decomposition before generation and on ambiguity mitigation through agent debate and self-reflection. Regarding RQ4, Class diagrams and Architecture/Component models concentrate the produced artifacts (25.0% each), with PlantUML as the dominant representation language, though roughly a quarter of studies report limitations in mapping dynamic relations in behavioral diagrams. Finally, the mapping highlights gaps in semantic validation, explainability, and modeling-specific benchmarks, each acknowledged within the corpus itself (P26, P36, P34) [14, 19, 28].

As future work, we highlight: the development of benchmarks and datasets dedicated to the semantic – and not only syntactic – evaluation of AI-generated models; the investigation of Explainable AI (XAI) mechanisms applied to the auditing of modeling decisions; and comparative studies specifically targeting the precision of dynamic relation mapping (directionality, cardinality, state transitions) in behavioral diagrams.

A further gap concerns language coverage: none of the 52 included studies evaluates model generation from Portuguese-language requirements or employs Brazilian-developed LLMs (e.g., Sabiá/Maritaca), an unexplored direction with direct relevance to research and industry communities operating in Portuguese.

Artifact Availability

The mapping artifacts (search results, screening decisions per criterion, structured extraction forms, and the verified BibTeX bibliography) are publicly available in an open repository: https://doi.org/10.5281/zenodo.21905779.

Acknowledgements

In accordance with ISE 2026 policy, the authors disclose the use of generative AI tools during manuscript preparation, including for language editing and text refinement, and take full responsibility for all content presented.

References

[1] Seif Abukhalaf, Mohammad Hamdaqa, and Foutse Khomh. 2023. On Codex Prompt Engineering for OCL Generation: An Empirical Study. In *IEEE Xplore*. 148–157. doi:10.1109/MSR59073.2023.00033

[2] Seif Abukhalaf, Mohammad Hamdaqa, and Foutse Khomh. 2024. PathOCL: Path-Based Prompt Augmentation for OCL Generation with GPT-4. *Proceedings - 2024 IEEE/ACM 1st International Conference on AI Foundation Models and Software Engineering, FORGE 2024* (2024), 108 – 118. doi:10.1145/3650105.3652290

[3] Bilal Al-Ahmad, Anas Alsobeh, Omar Meqdadi, and Nazimuddin Shaikh. 2025. A Student-Centric Evaluation Survey to Explore the Impact of LLMs on UML Modeling. *Information (Switzerland)* 16 (2025). doi:10.3390/info16070565

[4] Averi Bates, Ryan Vavricka, Shane Carleton, Ruosi Shao, and Chongle Pan. 2025. Unified modeling language code generation from diagram images using multimodal large language models. *Machine Learning with Applications* 20 (2025), 100660. doi:10.1016/j.mlwa.2025.100660

[5] Marco Calamo, Massimo Mecella, and Monique Snoeck. 2025. Assessing the Suitability of Large Language Models in Generating UML Class Diagrams as Conceptual Models. *Lecture Notes in Business Information Processing* 558 LNBIP (2025), 211 – 226. doi:10.1007/978-3-031-95397-2_13

[6] Sebastian Copei, Oliver Hohlfeld, Jens Kosiol, and Aleksandar Ristoski. 2025. LLMs Choose the Right Stack: From Patterns to Tools. In *IEEE Xplore*. 276–283. doi:10.1109/ASEW67777.2025.00057

[7] Gabriele De Vito, Fabio Palomba, Carmine Gravino, Sergio Di Martino, and Filomena Ferrucci. 2023. ECHO: An Approach to Enhance Use Case Quality Exploiting Large Language Models. In *IEEE Xplore*. 53–60. doi:10.1109/SEAA60479.2023.00017

[8] David Delgado, Lola Burgueño, and Robert Clarisó. 2026. A framework for assessing the capabilities of code generation of constraint domain-specific languages with large language models. *Journal of Systems and Software* 238 (2026), 112871. doi:10.1016/j.jss.2026.112871

[9] Damaris Naomi Dolha and Robert Andrei Buchmann. 2024. Experiments with natural language queries on RDF vs. XML-serialized BPMN diagrams. *Procedia Computer Science* 246 (2024), 3246–3255. doi:10.1016/j.procs.2024.09.315

[10] Tobias Eisenreich, Nicholas Friedlaender, and Stefan Wagner. 2025. Leveraging Large Language Models for Use Case Model Generation from Software Requirements. *Proceedings - 2025 40th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASEW 2025* (2025), 221 – 227. doi:10.1109/ASEW67777.2025.00050

[11] Zobia Erum and Issam Al-Azzoni. 2025. A GPT for Ecore Model Building. In *IEEE Xplore*. 54–57. doi:10.1109/FLLM67465.2025.11391086

[12] Matteo Esposito, Xiaozhou Li, Sergio Moreschini, Noman Ahmad, Tomas Cerny, Karthik Vaidhyanathan, Valentina Lenarduzzi, and Davide Taibi. 2026. Generative AI for software architecture. Applications, challenges, and future directions. *Journal of Systems and Software* 231 (2026), 112607. doi:10.1016/j.jss.2025.112607

[13] Alessio Ferrari, Sallam Abualhaijal, and Chetan Arora. 2024. Model Generation with LLMs: From Requirements to UML Sequence Diagrams. *Proceedings - 32nd IEEE International Requirements Engineering Conference Workshops, REW 2024* (2024), 291 – 300. doi:10.1109/REW61692.2024.00044

[14] Marios Fokaefs, Hashim Khan, and Borna Ahmadzadeh. 2025. Can AI Build Systems? an Exploratory Study on Generating Software Architecture With LLMS. *Proceedings - 2025 IEEE International Conference on Collaborative Advances in Software and Computing, CASCON 2025* (2025), 399 – 408. doi:10.1109/CASCON66301.2025.00069

[15] Carina Fresemann, Claudius Ellsel, Carl-Philipp Grunenwald, and Rainer Stark. 2025. Using Large Language Models for System Modeling: Current State and Future Implications. In *IEEE Xplore*. 1–8. doi:10.1109/ISSE65546.2025.11369978

[16] Giacomo Garaccione, Diego Maria Calabrese, Riccardo Coppola, and Luca Ardito. 2025. A comparison of different Large Language Models for the generation of UML class diagrams. In *IEEE Xplore*. 541–545. doi:10.1109/MODELS-C68889.2025.00078

[17] Stefana Gheorghita, Cosmin-Iulian Irimia, and Adrian Iftene. 2025. Automating Software Diagram Generation with Large Language Models. *Procedia Computer Science* 270 (2025), 713–722. doi:10.1016/j.procs.2025.09.191

[18] Ahmad Hatahet, Christoph Knieke, and Andreas Rausch. 2025. Generating Software Architecture Description from Source Code using Reverse Engineering and Large Language Model. *Proceedings - 2025 ACM/IEEE 28th International Conference on Model Driven Engineering Languages and Systems Companion, MODELS-C 2025* (2025), 566 – 575. doi:10.1109/MODELS-C68889.2025.00080

[19] Dongming Jin, Zhi Jin, Xiaohong Chen, and Chunhui Wang. 2024. ChatModeler: A human-machine collaborative and iterative requirements elicitation

and modeling approach via large language models; [ChatModeler]. *Jisuanji Yanjiu yu Fazhan/Computer Research and Development* 61 (2024), 338 – 350. doi:10.7544/issn1000-1239.202330746

[20] Dae-Kyoo Kim. 2025. Quantitative Assessment of Generative Large Language Models on Design Pattern Application. *Computers, Materials and Continua* 82 (2025), 3843–3872. doi:10.32604/cmc.2025.062552

[21] Dae-Kyoo Kim. 2026. Artifact validity under varying agent configurations in LLM-assisted software development: A comparative analysis. *Information and Software Technology* 192 (2026), 108022. doi:10.1016/j.infsof.2026.108022

[22] Barbara Kitchenham and Stuart Charters. 2007. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Technical Report EBSE-2007-01. Keele University and University of Durham.

[23] Yuan Kong, Nan Zhang, Zhenhua Duan, and Bin Yu. 2025. Collaboration with Generative AI to improve Requirements Change. *Computer Standards & Interfaces* 94 (2025), 104013. doi:10.1016/j.csi.2025.104013

[24] Victor Lamas, Daniel Garcia-Gonzalez, Luca Sala, and Miguel R. Luaces. 2026. DSL-Xpert 2.0: Enhancing LLM-driven code generation for domain-specific languages. *Information and Software Technology* 190 (2026), 107954. doi:10.1016/j.infsof.2025.107954

[25] Josip Tomo Licardo, Nikola Tanković, and Darko Etinger. 2024. A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing. *Procedia Computer Science* 239 (2024), 483–490. doi:10.1016/j.procs.2024.06.196

[26] Xianhui Liu, Yueying Liu, Yiheng Zhuang, and Wenlong Hou. 2026. UCD-LLM: A use case diagram requirement modeling multi-agent framework with large language model. *Information and Software Technology* 190 (2026), 107955. doi:10.1016/j.infsof.2025.107955

[27] Yue Liu, Sin Kit Lo, Qinghua Lu, Liming Zhu, Dehai Zhao, Xiwei Xu, Stefan Harrer, and Jon Whittle. 2025. Agent design pattern catalogue: A collection of architectural patterns for foundation model based agents. *Journal of Systems and Software* 220 (2025), 112278. doi:10.1016/j.jss.2024.112278

[28] Nikolaos Malamas, Emmanouil Tsardoulias, Konstantinos Panayiotou, and Andreas L. Symeonidis. 2025. Toward efficient vibe coding: An LLM-based agent for low-code software development. *Journal of Computer Languages* 85 (2025), 101367. doi:10.1016/j.cola.2025.101367

[29] João José Maranhão and Eduardo Martins Guerra. 2024. A Prompt Pattern Sequence Approach to Apply Generative AI in Assisting Software Architecture Decision-making. *ACM International Conference Proceeding Series* (2024). doi:10.1145/3698322.3698324

[30] Christoph Mayr-Dorn, Anmol Bilal, Cosmina Ratiu, and Alexander Egyed. 2025. Generating Quality Assurance Constraints From Natural Language With LLMs. *Journal of Software: Evolution and Process* 37 (2025). doi:10.1002/smr.70062

[31] Lukasz Mazur, Nenad Petrovic, James Pontes Miranda, Ansgar Radermacher, Robert Rasche, and Alois Knoll. 2025. Querying Large Automotive Software Models: Agentic vs. Direct LLM Approaches. *2025 2nd International Generative AI and Computational Language Modelling Conference, GACLM 2025* (2025), 221 – 228. doi:10.1109/GACLM67198.2025.11232128

[32] Jonathan Silva Mercado, Qin Ma, Sybren de Kinderen, Karolin Winter, and Jordi Cabot. 2026. CLERK: A Companion Large Language Model Expert for modeling Regulatory Knowledge. *Data & Knowledge Engineering* 164 (2026), 102578. doi:10.1016/j.datak.2026.102578

[33] Vittoriano Muttillo, Claudio Di Sipio, Riccardo Rubei, and Luca Berardinelli. 2025. Leveraging synthetic trace generation of modeling operations for intelligent modeling assistants using large language models. *Information and Software Technology* 186 (2025), 107806. doi:10.1016/j.infsof.2025.107806

[34] Vittoriano Muttillo, Claudio Di Sipio, Riccardo Rubei, Luca Berardinelli, and MohammadHadi Dehghani. 2024. Towards Synthetic Trace Generation of Modeling Operations using In-Context Learning Approach. *Proceedings - 2024 39th ACM/IEEE International Conference on Automated Software Engineering, ASE 2024* (2024), 619 – 630. doi:10.1145/3691620.3695058

[35] Lahbib Naimi, El Mahi Bouziane, Abdeslam Jakimi, Rachid Saadane, and Abdellah Chehri. 2024. Automating Software Documentation: Employing LLMs for Precise Use Case Description. *Procedia Computer Science* 246 (2024), 1346 – 1354. doi:10.1016/j.procs.2024.09.568

[36] Reshma P. Nair, M. G. Thushara, and Vijayan Sugumaran. 2025. Fine-Tuned LLMs Versus Rule-Based NLP for UML Diagram Generation: An Educational Evaluation. *IEEE Access* 13 (2025), 211605–211619. doi:10.1109/ACCESS.2025.3638372

[37] Benjamin Nast, Leon Görgen, Eric Müller, Marcus Triller, and Kurt Sandkuhl. 2025. Exploring large language models in enterprise modeling. *Discover Artificial Intelligence* 5 (2025). doi:10.1007/s44163-025-00585-2

[38] Van-Viet Nguyen, Huu-Khanh Nguyen, Kim-Son Nguyen, Thi Minh-Hue Luong, Duc-Quang Vu, Trung-Nghia Phung, and The-Vinh Nguyen. 2026. A Novel Unified Framework for Automated Generation and Multimodal Validation of UML Diagrams. *CMES - Computer Modeling in Engineering and Sciences* 146 (2026). doi:10.32604/cmes.2025.075442

[39] Oluwafemi Odu, Alvine B. Belle, Song Wang, Segla Kpodjedo, Timothy C. Lethbridge, and Hadi Hemmati. 2025. Automatic instantiation of assurance cases from patterns using large language models. *Journal of Systems and Software* 222 (2025), 112353. doi:10.1016/j.jss.2025.112353

[40] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, and David Moher. 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *BMJ* 372 (2021), n71. doi:10.1136/bmj.n71

[41] Krish Patel, Brian Sheehan, and Shamsnaz Bhada. 2025. Exploring Intelligent Document Processing Technology for Discovering Requirements Debt in Systems Engineering Management Plans. *Procedia Computer Science* 268 (2025), 437–446. doi:10.1016/j.procs.2025.08.223

[42] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*. 68–77.

[43] Gokul Rejithkumar, Preethu Rose Anish, Jyoti Shukla, and Smita Ghaisas. 2024. Probing with Precision: Probing Question Generation for Architectural Information Elicitation. *Proceedings - 2024 IEEE/ACM Workshop on Multi-disciplinary, Open, and RElevant Requirements Engineering, MO2RE 2024* (2024), 8 – 14. doi:10.1145/3643666.3648577

[44] Stefano Rizzi, Matteo Francia, Enrico Gallinucci, and Matteo Golfarelli. 2025. Conceptual design of multidimensional cubes with LLMs: An investigation. *Data & Knowledge Engineering* 159 (2025), 102452. doi:10.1016/j.datak.2025.102452

[45] Satrio Adi Rukmono, Lina Ochoa, and Michel R.V. Chaudron. 2023. Achieving High-Level Software Component Summarization via Hierarchical Chain-of-Thought Prompting and Static Code Analysis. In *IEEE Xplore*. 7–12. doi:10.1109/ICoDSE59534.2023.10292037

[46] Satrio Adi Rukmono, Lina Ochoa, and Michel R.V. Chaudron. 2024. Deductive Software Architecture Recovery via Chain-of-thought Prompting. *Proceedings - International Conference on Software Engineering* (2024), 92 – 96. doi:10.1145/3639476.3639776

[47] Hanan Abdulwahab Siala and Kevin Lano. 2026. Leveraging LLMs for abstracting UML and OCL representations from Java and Python programs. *Journal of Systems and Software* 238 (2026), 112874. doi:10.1016/j.jss.2026.112874

[48] Ruoyu Su, Noman Ahmad, Matteo Esposito, Andrea Janes, Davide Taibi, and Valentina Lenarduzzi. 2026. Emerging trends in software architecture from the practitioner's perspective: A five-year review. *Journal of Systems and Software* 236 (2026), 112820. doi:10.1016/j.jss.2026.112820

[49] Mohammad Tanhaei. 2026. MedNegotiator: Automating requirements negotiation in healthcare; simulating clinical and technical perspectives using Generative AI agents. *Array* 30 (2026), 100783. doi:10.1016/j.array.2026.100783

[50] Mircea Trifan, Bogdan Ionescu, and Dan Ionescu. 2025. Generative AI for Diagrams as Code and Code as Diagrams. In *IEEE Xplore*. 01–06. doi:10.1109/SACI66288.2025.11030105

[51] Francesco Vitale, Francesco Flammini, Mauro Caporuscio, and Nicola Mazzocca. 2026. Architecting software monitors for control-flow anomaly detection through large language models and conformance checking. *Information and Software Technology* 195 (2026), 108133. doi:10.1016/j.infsof.2026.108133

[52] Bangrui Wan, Zheng Wei, Shiyu Wang, Jiangping Huang, and Chunqiang Hu. 2026. Structure-guided function-level code generation with LLMs via UML activity diagrams. *Neurocomputing* 669 (2026), 132502. doi:10.1016/j.neucom.2025.132502

[53] Chong Wang, Beian Wang, Peng Liang, and Jie Liang. 2026. Assessing UML diagrams by GPT: Implications for education. *Journal of Systems and Software* 234 (2026), 112709. doi:10.1016/j.jss.2025.112709

[54] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.

[55] Yongheng Zhang, Tingwen Du, Yunshan Ma, Xiang Wang, Yi Xie, Guozheng Yang, Yuliang Lu, and Ee-Chien Chang. 2025. AttacKG+: Boosting attack graph construction with Large Language Models. *Computers & Security* 150 (2025), 104220. doi:10.1016/j.cose.2024.104220

[56] Zelong Zhao, Nan Zhang, Bin Yu, and Zhenhua Duan. 2024. Generating Java code pairing with ChatGPT. *Theoretical Computer Science* 1021 (2024), 114879. doi:10.1016/j.tcs.2024.114879

[57] Rui Zhu, Jiapeng Chen, Tianrui Bai, Hua Yue, Jianglong Qin, and Xuan Zhang. 2025. MASFlow: Multi-Agent Based Service Workflow Generation. *Proceedings - 2025 IEEE International Conference on Software Services Engineering, SSE 2025* (2025), 24 – 30. doi:10.1109/SSE67621.2025.00012

[58] Zhixian Zhuang, Xiaodong Lee, Aiyao Zhang, Jiuqi Wei, Yufan Fu, and Botao Peng. 2026. Structured policy modeling and context-aware generation for multi-jurisdictional compliance in global software systems. *Information and Software Technology* 193 (2026), 108041. doi:10.1016/j.infsof.2026.108041

[59] Yaxin Zou, Zhibin Yang, Hao Liu, Jiawei Liang, Yong Zhou, and Zonghua Gu. 2025. Automated AADL Architecture Modeling: Leveraging Large Language Models for Safety-Critical Software. In *IEEE Xplore*. 311–321. doi:10.1109/MODELS-C68889.2025.00050