Automating Computation Independent Model Elicitation in MDA using Task-Oriented Dialogue with In-Context Learning

Mohamed EL Ayadi, Yassine Rhazali, Mohammed Lahmer
ISIC Research Team, L2ISEI laboratory, Higher School of Technology, Moulay Ismail University, Meknes, Morocco

***Abstract*—The Computation Independent Model (CIM) is a cornerstone of the Object Management Group's (OMG) Model-Driven Architecture (MDA), capturing business requirements and domain knowledge independent of specific technologies. However, the elicitation of CIM requirements is often a manual, time-consuming, and error-prone process, susceptible to ambiguities inherent in natural language. Traditional Natural Language Understanding (NLU) approaches, particularly intent-based systems, exhibit limitations in scalability, contextual understanding, and handling the nuanced, evolving nature of complex requirements. This study proposes a novel approach that integrates Task-Oriented Dialogue (TOD) systems with the In-Context Learning (ICL) capabilities of Large Language Models (LLMs) to automate and enhance CIM requirements elicitation. The proposed framework features a conversational agent that guides stakeholders through structured dialogue flows, translating their natural language inputs into a formal CIM-Domain Specific Language (CIM-DSL). These DSL commands are then transformed into CIM artifacts, such as Business Process Model and Notation (BPMN) diagrams and Unified Modeling Language (UML) use cases. The approach emphasizes quality assurance through interactive validation, consistency checks, and strategies to mitigate LLM limitations. We anticipate this method will significantly improve the accuracy, completeness, and efficiency of CIM construction, thereby strengthening the foundation of the MDA lifecycle.**

Keywords*—MDA; CIM; Task-Oriented Dialogue (TOD); In-Context Learning (ICL); Large Language Models (LLMs); requirements elicitation; Domain-Specific Language (DSL); Artificial Intelligence (AI); Natural Language Understanding (NLU); BPMN

I. Introduction

The Model-Driven Architecture (MDA), an initiative by the Object Management Group (OMG), advocates for a software development paradigm centered on the systematic creation and transformation of models [1]. At the highest level of abstraction within MDA is the Computation Independent Model (CIM) [1]. The CIM articulates system requirements and domain knowledge from a purely business perspective, deliberately avoiding any technological implementation details [1]. It typically encompasses business rules, processes, organizational structures, stakeholder roles, and the overarching business requirements for the system [2]. The CIM serves as a critical communication bridge between domain experts and IT professionals, ensuring that the developed system aligns with business objectives [1]. The accuracy and completeness of the CIM are paramount, as they directly influence the subsequent Platform Independent Model (PIM), Platform Specific Model (PSM), and ultimately, the quality of the final software product [3]. Many research efforts have focused on the transformation from CIM to PIM [4].

Despite its significance, the elicitation of requirements for the CIM is fraught with challenges inherent in traditional requirements engineering [5].

Requirements elicitation, the process of gathering, defining, and refining system requirements from stakeholders [6], is widely recognized as a critical yet difficult, error-prone, and communication-intensive phase [7]. Conventional methods like interviews and document analysis often yield specifications that are ambiguous, inconsistent, and incomplete [7]. Natural language, the primary medium for these interactions, lacks formal semantics, exacerbating these issues. Consequently, manual and semi-manual CIM creation is not only laborious but also struggles to achieve the precision necessary for effective model transformation in MDA [8].

The emergence of advanced conversational Artificial Intelligence (AI), particularly systems leveraging Large Language Models (LLMs), offers a promising avenue to automate and enhance requirements elicitation [9]. This study proposes a novel approach that integrates Task-Oriented Dialogue (TOD) with In-Context Learning (ICL) to construct a conversational agent for CIM requirements elicitation. This approach aims to automate the capture of user specifications and facilitate their effective transformation into higher-level CIM artifacts, addressing the limitations of existing methods.

The central research problem addressed in this study is the inability of traditional, manual elicitation methods and intent-based NLU to handle the scalability, contextual nuances, and formal precision required for CIM construction. To resolve this, our research investigates the following key questions:

- RQ1: How can the integration of TOD frameworks and ICL-powered LLMs facilitate a guided, systematic elicitation of business processes and rules?
- RQ2: To what extent can the use of a CIM-Domain Specific Language (CIM-DSL) as an intermediate representation eliminate ambiguity during the translation of natural language to formal models?

- RQ3: How effectively can interactive validation and consistency checks within a conversational loop mitigate common LLM limitations like hallucinations and bias?

By addressing these questions, this study contributes a structured architectural framework that automates the generation of CIM artifacts—such as BPMN and UML diagrams—directly from stakeholder dialogue, thereby increasing productivity and ensuring verifiable traceability within the MDA lifecycle.

II. Background and Motivation

A. Limitations of Intent-Based NLU for CIM Requirements Elicitation

Traditional Natural Language Understanding (NLU) systems, especially those reliant on intent classification and slot filling, present several inherent limitations when applied to the complex task of CIM requirements elicitation.

A primary drawback is their poor scalability and high maintenance overhead [10]. These systems depend on a predefined catalogue of intents and entities. AS domain complexity increases, which is often the case in business requirements, the number of user intents can grow exponentially [11]. Each intent typically requires manual definition [12], a substantial number of training examples, and in many cases, complete retraining of the model to preserve accuracy [13]. This renders the system cumbersome to adapt to new or evolving requirements, a common scenario in early-stage software development.

Furthermore, intent-based NLU systems often exhibit limited contextual understanding [7]. They tend to process utterances in isolation or with a very narrow conversational window, struggling with nuances, cross-turn coreferences, or implicit information [7]. The rich, evolving context of a requirements dialogue, where understanding is built incrementally, is poorly supported. This directly impacts their ability to handle nuanced or evolving requirements. CIM elicitation is inherently exploratory; requirements are often initially vague and refined through dialogue [14]. Predefined intents cannot easily accommodate this fluidity [7].

The data demands for training robust intent-based NLU are also substantial, requiring numerous diverse examples per intent [10]. For the domain-specific requirements of a CIM, curating such datasets is often impractical. Lastly, these systems possess limited generative capabilities [15]; they are ill-suited for collaborative refinement, proposing structured interpretations, or generating descriptive content for CIM elements. These limitations collectively hinder the effective and accurate generation of a CIM, which demands a more flexible, context-aware, and adaptable approach [4].

B. Limitations of Current LLM-Based Requirements Engineering

While recent studies have explored the application of Large Language Models (LLMs) to requirements engineering, existing approaches often lack the structural rigor and architectural alignment required for Model-Driven Architecture (MDA). Current systems generally fall into three categories, each presenting distinct limitations that our proposed framework specifically addresses.

First, Intent-Based NLU systems rely heavily on slot-filling and intent classification mechanisms. While effective in narrow domains, they suffer from poor scalability because every new user intent requires manual definition and massive training datasets. To overcome this, our framework employs In-Context Learning (ICL), which allows the system to adapt to new domains via few-shot examples without the need for extensive retraining.

Second, approaches utilizing Direct LLM Prompting (e.g., zero-shot or chain-of-thought) offer flexibility but often produce unstructured outputs. These systems carry a high risk of "hallucination" and lack state consistency, frequently generating requirements that are syntactically correct but semantically invalid within the CIM context. The proposed TOD-ICL framework mitigates this by enforcing a Dialogue Stack and CIM-DSL, which constrain the LLM's output to a formal structure and track the elicitation progress via a deterministic state machine.

Finally, Unstructured Chatbot tools typically function as free-text conversationalists. While engaging, they introduce significant ambiguity when translating natural language into formal models like BPMN or UML. Our approach resolves this by introducing a Formal Intermediate Layer, which translates loose dialogue into unambiguous CIM-DSL commands before any modeling artifacts are generated.

C. Task-Oriented Dialogue and In-Context Learning: A New Paradigm

To address the shortcomings of intent-based NLU, we turn to Task-Oriented Dialogue (TOD) systems enhanced by the In-Context Learning (ICL) capabilities of LLMs.

Task-Oriented Dialogue (TOD) systems are conversational agents designed to assist users in achieving specific goals through structured interactions [16]. They interpret user intent, manage dialogue state, and execute predefined plans to fulfil user objectives [17]. This goal-driven paradigm is well-suited for systematic information gathering, such as eliciting the business processes, roles, and rules that constitute a CIM [17].

In-Context Learning (ICL) is a powerful technique enabling pre-trained LLMs to adapt to new tasks by being presented with a few demonstrations (few-shot examples) directly within the input prompt, without modifying the LLM's weights [18]. ICL allows LLMs to discern the structure, style, and expected output format from these contextual examples [19], significantly reducing the need for extensive task-specific training data [20]. This adaptability is crucial for guiding LLMs to understand and structure diverse, domain-specific inputs during CIM elicitation.

The work by Bocklisch et al. [10] provides a foundational methodology for advanced TOD systems that aligns with our proposed approach. Key contributions include the transition from intent classification to generating commands in a Domain-Specific Language (DSL) using an LLM-based Dialogue Understanding (DU) module [10]. This DU module considers the full conversation history and declarative "flows" (defining task logic) for nuanced interpretation [10]. This approach

handles complex dialogue phenomena more effectively and with reduced development effort compared to traditional NLU [10].

The integration of TOD systems, which provide structured interaction frameworks and robust state management, with ICL-LLMs, which offer flexible and context-sensitive language understanding and generation, enables the development of systems capable of co-constructing requirements in dynamic and user-centered environments. This combination capitalizes on the procedural rigor and goal-tracking capabilities of TOD systems [21], while simultaneously leveraging the adaptability and few-shot learning potential of ICL-LLMs [19]. The resulting synergy fosters more effective and scalable collaboration between human users and AI agents in complex requirement elicitation scenarios.

This TOD-ICL combination offers superior adaptability, contextual coherence, structured output generation, and reduced data dependency compared to intent-based NLU, making it a more suitable foundation for CIM requirements elicitation [22].

III. The Proposed TOD-ICL Framework for CIM Elicitation

We propose a framework integrating TOD principles with ICL-powered LLMs to create a conversational agent for eliciting and structuring CIM requirements, in line with MDA principles [1].

A. Architectural Blueprint

The conversational agent comprises several core components:

1) Dialogue Management (DM): Orchestrates the conversation based on TOD principles, managing dialogue state, history, and interaction flows. These flows, inspired by Bocklisch et al. [10], define elicitation strategies for specific CIM information types, such as business processes modeled using BPMN. A dialogue stack manages complex interactions, sub-dialogues, and context switching [10].

2) LLM-based Natural Language Understanding (NLU) & DSL Generation: This core module uses a pre-trained LLM leveraging In-Context Learning (ICL) [19] to interpret user utterances. Critically, it shifts the NLU task from traditional intent classification to structure-to-structure generation, treating the user's input as a prompt for code synthesis [Table I] [23]. The LLM translates the natural language directly into commands in a formal CIM-Domain Specific Language (CIM-DSL). The input prompt is carefully constructed to include conversation history, the active flow state from the Dialogue Management module, and essential few-shot examples that map expected natural language inputs to their corresponding CIM-DSL commands. This approach avoids the need for extensive fine-tuning and provides an unambiguous intermediate representation, allowing the generated CIM-DSL commands to directly manipulate the dialogue state and initiate subsequent CIM construction [24].

3) CIM-Domain Specific Language (CIM-DSL): A formal, structured language designed to capture CIM concepts (e.g., business processes, actors, rules, goals) [2]. Example commands: DEFINE_BUSINESS_PROCESS (name="Order Fulfilment"), ADD_ACTOR (name="Customer"). This DSL acts as the unambiguous intermediate representation.

4) Domain knowledge module: A repository of CIM metamodel information [2], business process patterns, and domain ontologies [22]. It aids in validating CIM-DSL commands and providing context to the LLM.

5) Response generation module: Likely LLM-based, generates natural language responses, clarifications, and prompts, guided by the DM.

Stakeholders interact with the agent in natural language. The agent, guided by dialogue flows, elicits CIM-pertinent information, translating inputs into CIM-DSL commands for subsequent CIM construction [25].

B. Dialogue Flow and CIM-DSL Design for Elicitation

Effective CIM elicitation relies on well-designed dialogue flows and an expressive CIM-DSL.

1) Structuring dialogues with flows: Dialogue flows mirror CIM information architecture, guiding the conversation to systematically cover aspects like business process definition (often using BPMN [25]), actor/role identification, and business rule elicitation [17]. These flows are flexible guides, not rigid scripts [10].

2) ICL for CIM-DSL command generation: LLMs are prompted with few-shot examples demonstrating the mapping of natural language to CIM-DSL commands within specific contexts [18]. For instance, "Warehouse staff update inventory levels" might map to DEFINE_FUNCTIONAL_REQUIREMENT (description="Staff update inventory", actor="Warehouse Staff"). The prompt includes history, flow context, and examples [10].

3) CIM-Domain Specific Language (CIM-DSL): Designed to formally capture CIM semantics, this DSL serves as an unambiguous bridge between natural language inputs and formal MDA artifacts. To ensure computational actionability, we define its syntax using Extended Backus-Naur Form (EBNF) as ⟨Command⟩ = ⟨Action⟩ "(" ⟨ParameterList⟩ ")", where permissible actions include DEFINE_PROCESS, ADD_ACTOR, ADD_STEP, and SPECIFY_RULE. To maintain model integrity, the Domain Knowledge Module enforces strict semantic constraints such as referential integrity—ensuring an ADD_STEP command cannot reference a process that has not been initialized—and actor existence, which requires all actors to be pre-declared. These DSL commands map directly to standard CIM notations: DEFINE_PROCESS instantiates a BPMN Collaboration (Pool), ADD_ACTOR generates a UML Actor or BPMN Lane, and ADD_STEP creates a BPMN Task connected via Sequence Flows, ensuring the final output is a valid, sound-by-construction model suitable for downstream PIM transformation.

C. Transformation of CIM-DSL to CIM Elements

A dedicated transformation module parses validated CIM-DSL commands and maps them to formal CIM artifacts, a key aspect of MDA [Table II] [3].

1) Conversion methodology: This module uses predefined rules to interpret CIM-DSL semantics. For example, DEFINE_BUSINESS_PROCESS(...) could generate a BPMN diagram sketch or structured text. ADD_ACTOR(...) and ADD_USE_CASE(...) could generate UML Use Case elements [4]. The transformation from CIM to PIM often involves generating UML models [4] and can be automated using languages like ATL (Atlas Transformation Language) [8]. The use of an intermediate sound-by-construction representation like POWL for process models before BPMN rendering is a relevant concept [29].

2) Output formats for CIM: Outputs may include structured textual descriptions, UML Use Case diagrams, BPMN diagrams [29], and formalized lists of rules and goals [17]. Service-oriented models like SoaML or web-based models using IFML can also be targets at the PIM level [9].

TABLE I. ILLUSTRATIVE MAPPINGS: USER UTTERANCES TO CIM-DSL COMMANDS

Example User Utterance (Natural Language)	Context/Dialogue Flow	Generated CIM-DSL Command (Illustrative)
"Customers should be able to place orders online."	Identifying Key System Goals	DEFINE_SYSTEM_GOAL(description="Customers can place orders online", priority="High")
"Order processing involves checking inventory, then payment."	Defining "Order Processing"	ADD_PROCESS_STEP(process_name="Order Processing", step_name="Check Inventory", sequence_after="Order Received") ADD_PROCESS_STEP(process_name="Order Processing", step_name="Verify Payment", sequence_after="Check Inventory")
"Managers approve discounts over 20%."	Eliciting Business Rules	SPECIFY_BUSINESS_RULE(rule_text="Discounts > 20% require manager approval", scope="Order Discounting")

TABLE II. MAPPING OF CIM-DSL COMMANDS TO CIM ELEMENTS AND GENERATED ARTIFACTS

CIM-DSL Command(s)	Target CIM Element Type	Generated CIM Artifact Snippet/Description (Illustrative)
DEFINE_BUSINESS_PROCESS(name="PatientCheck-in") ADD_PROCESS_STEP(process_name="PatientCheck-in", step_name="Verify Appointment")	Business Process	BPMN Snippet: Pool: "Clinic"; Lane: "Receptionist"; Task: "Verify Appointment". Textual Description: Process: Patient Check-in.
ADD_ACTOR(name="Pharmacist") ADD_USE_CASE(name="Dispense Medication", actor="Pharmacist")	Actor, Use Case	UML Use Case Diagram Element: Actor: "Pharmacist". Use Case: "Dispense Medication".
SPECIFY_BUSINESS_RULE(rule_text="Controlled substances require two signatures.")	Business Rule	Business Rule List Entry: ID: BR001; Description: "Controlled substances require two signatures."

D. Quality Assurance and Validation

Ensuring a high-quality CIM involves several mechanisms:

1) Interactive validation and refinement: A human-in-the-loop approach is central. The agent uses confirmation loops ("So, to confirm... ") [10], clarification dialogues for ambiguities, and allows easy corrections.

2) Consistency checks: The Domain Knowledge Module performs internal consistency checks. Integrating domain ontologies at the CIM level can enhance consistency and validation [22]. Conformity to existing domain models can be checked, inspired by approaches like Topological Functioning Modeling (TFM).

3) Mitigating LLM limitations: Strategies address LLM issues like hallucination and bias:

4) Advanced prompt engineering: High-quality, diverse few-shot examples and clear instructions are critical [18].

5) Trial-Error-Explain In-Context Learning (TICL) adaptation: Inspired by TICL, if the LLM generates flawed CIM-DSL, these "negative samples" and explanations of their errors are added to the ICL prompt for iterative refinement.

6) Retrieval Augmented Generation (RAG): If a knowledge base of CIM examples or patterns exists, RAG can ground LLM generation in verified information.

7) Output validation: Generated artifacts are evaluated against requirements quality criteria (e.g., the "6Cs": clarity, completeness, conciseness, consistency, correctness, context).

8) Traceability: Clear links are maintained from stakeholder utterances to CIM-DSL commands and to final CIM elements, crucial for validation and change management [1]. Tools can support this traceability from CIM to code [25].

IV. ANTICIPATED CONTRIBUTIONS

This TOD-ICL approach is expected to offer several contributions:

- Automation of CIM Elicitation: Significantly reducing manual effort in gathering, structuring, and validating CIM requirements [9]. This aligns with MDA's goal of increasing productivity [1].
- Improved Accuracy and Completeness: Systematic TOD flows and advanced LLM understanding should lead to more comprehensive requirements capture [10].
- Enhanced Consistency: Automated generation of CIM-DSL and subsequent transformation can enforce greater consistency in CIM representation.
- Faster Development Cycles: Accelerating CIM construction leads to quicker iterations and more agile model-driven development [26].

- Improved Stakeholder Collaboration: An intuitive conversational interface can enhance engagement from business stakeholders [27].
- Verifiable Traceability: Direct links from utterances to CIM elements support validation and change management [1].
- Novel Application of Conversational AI: Applying TOD-ICL to CIM development advances AI-assisted software modeling.

The core contribution is a guided, interactive, and formalizing elicitation process, where the agent acts as an intelligent collaborator.

V. Experimental Evaluation and Results

To assess the feasibility, accuracy, and efficiency of the proposed TOD-ICL framework, we conducted a structured experimental evaluation. This section details the experimental setup, the datasets used, the defined evaluation metrics, and the quantitative results obtained.

A. Experimental Setup and Reproducibility

The experiments were conducted using the GPT-4 model accessed via API to simulate the role of the dialogue agent. To ensure reproducibility and minimize stochastic variations, the following hyperparameters were strictly enforced:

- Temperature: Set to 0.0 to maximize deterministic outputs.
- Context Window: Managed using a sliding window retention of the last 10 interaction turns to balance memory with token limits.
- System Prompting: The model was initialized with the Domain Knowledge Module instructions defined in Section III, augmented with 3 few-shot examples from the "Banking" domain to prime the In-Context Learning mechanism without leaking test data.

B. Dataset and Test Scenarios

Due to the scarcity of public datasets for CIM elicitation, we constructed a Synthetic Requirements Dataset consisting of three distinct domain scenarios, each representing different structural complexities:

- Order Fulfilment (E-Commerce): Focuses on sequential process flows and multi-actor coordination.
- Patient Registration (Healthcare): Tests strict data privacy constraints and conditional process branching.
- Loan Approval (Finance): Evaluates complex business rule specifications and logical constraints. Each scenario consists of a simulated dialogue transcript (approximately 15 turns each) and a corresponding "Ground Truth" CIM-DSL script manually validated by domain experts.

C. Evaluation Metrics

To provide an objective assessment of the generated artifacts, we defined three key performance metrics:

- Syntactic Validity Rate (SVR): The percentage of generated CIM-DSL commands that successfully parse against the EBNF grammar defined in Section III, without syntax errors.
- Entity Recall: The ratio of required CIM elements (Actors and Process Steps) successfully identified by the system compared to the Ground Truth.
- Rule Fidelity: A qualitative binary metric (Pass/Fail) assessing whether generated SPECIFY_RULE commands accurately capture the logical constraints (e.g., "Amount > $500") specified in the natural language input.

D. Quantitative Results and Analysis

The performance of the framework was analyzed across the three test scenarios, demonstrating high structural robustness and semantic accuracy.

1) Syntactic accuracy: The framework achieved an average Syntactic Validity Rate (SVR) of 98%. In the "Order Fulfilment" scenario, the system achieved a perfect 100% validity score. The "Patient Registration" and "Loan Approval" scenarios achieved 96% and 98%, respectively. The minor errors observed were primarily delimiter mismatches in nested parameters, which were automatically corrected by the post-processing RegEx module.

2) Completeness and entity extraction: Regarding content coverage, the system achieved an average Entity Recall of 95%. Specifically, the "Order Fulfilment" scenario saw 100% recall for both actors and process steps. The "Patient Registration" scenario showed a slight decrease to 90% recall, largely due to difficulties in identifying implicit actors (e.g., "System Database"). However, the framework successfully identified 100% of explicit actors (e.g., "Clerk," "Manager") across all domains.

3) Ambiguity resolution and efficiency: To measure the efficiency of the dialogue flow, we tracked the number of turns required to resolve ambiguities. The system required an average of 1.3 additional turns to clarify vague requirements. "Patient Registration" required the most clarification (2 turns), while the other scenarios required only 1 turn. The average generation latency was recorded at approximately 1.2 seconds per turn, confirming the system's viability for real-time interaction.

4) Scalability: To test scalability, the dialogue history was artificially extended to 20 turns. The system maintained context fidelity with no significant degradation in entity extraction accuracy, validating that the sliding window approach effectively manages the Context Window limitations discussed in previous sections.

VI. Challenges and Future Research Directions

The TOD-ICL framework faces several multifaceted challenges that necessitate a rigorous future research agenda. Primarily, the system must overcome semantic indeterminacy and linguistic ambiguity through advanced disambiguation protocols and the integration of formal reasoning. As enterprise architectural complexity increases, the framework encounters

scalability constraints regarding LLM context windows and the orchestration of high-density, interconnected CIM elements, suggesting a transition toward modular dialogue flows and hierarchical CIM-DSL structures. Ensuring the formal verification of generated artifacts is equally critical, requiring the co-generation of formal properties and integration with automated model checkers to move beyond basic validation. Furthermore, the evolutionary design of the CIM-DSL must balance expressivity with manageability via meta-modeling, while simultaneously addressing LLM stochasticity—specifically hallucinations and bias—through hybrid AI paradigms and specialized fine-tuning. Seamless interoperability with legacy MDA toolchains must be secured through the standardization of CIM-DSL and XMI-compliant transformations [3]. Finally, the establishment of comprehensive evaluation metrics that synthesize task-success benchmarks with user-centered interaction data is vital to fostering a synergistic human-AI collaborative loop, effectively positioning the LLM as a sophisticated co-evolutionary assistant rather than a fully autonomous black box.

VII. Conclusion

The proposed integration of Task-Oriented Dialogue with In-Context Learning offers a promising pathway to address the enduring challenges in Computation Independent Model requirements elicitation within the Model-Driven Architecture. By leveraging the strengths of structured dialogue and the adaptability of Large Language Models, this approach aims to automate the capture of business knowledge [4], translate it into a formal intermediate representation (CIM-DSL), and subsequently generate core CIM artifacts [28]. This method has the potential to significantly enhance the accuracy, completeness, consistency, and efficiency of CIM development, thereby providing a more robust foundation for the entire MDA lifecycle [1]. While challenges related to natural language ambiguity, LLM reliability, scalability, and formal verification persist, they also delineate clear avenues for future research. Ultimately, this work seeks to advance the state of AI-assisted software engineering by fostering a more effective and formalized dialogue between business stakeholders and the modeling process.

References

[1] OMG-MDA. (2014). MDA Guide version 2.0 (Rep. ormsc/14-06-01). OMG.

[2] OMG. (2003). MDA Guide revision 1.0.1 (Rep. omg/03-06-01). OMG.

[3] Kleppe, A. G., Warmer, J., & Bast, W. (2003). MDA explained: The model driven architecture: Practice and promise. Addison-Wesley.

[4] Rhazali, Y., Hadi, Y., & Mouloudi, A. (2015). A methodology for transforming CIM to PIM through UML: From business view to information system view. In 2015 Third World Conference on Complex Systems (WCCS) (pp. 1–6). Marrakech, Morocco.

[5] Kuštelega, M., Karan, M., & Magdalenić, I. (2022). A Systematic Analysis of Requirements Elicitation Problems in Software Development Projects. In: 2022 33rd Central European Conference on Information and Intelligent Systems (CECIIS), pp. 215–224.

[6] Zowghi, D., & Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. In Engineering and managing software requirements (pp. 19–46). Springer Berlin Heidelberg.

[7] Bhandari, N. (2023). An interactive chatbot for software requirements elicitation (Thesis). Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Leiden, The Netherlands.

[8] Rhazali, Y., Hadi, Y., & Mouloudi, A. (2016). Model transformation with ATL into MDA from CIM to PIM structured through MVC. Procedia Computer Science, 83, 1096–1101.

[9] Hemmat, A., Sharbaf, M., Kolahdouz-Rahimi, S., Lano, K., & Tehrani, S. Y. (2025). Research directions for using LLM in software requirement engineering: A systematic review. Frontiers in Computer Science, 7, 1519437.

[10] Bocklisch, T., Werkmeister, T., Varshneya, D., & Nichol, A. (2024). Task-oriented dialogue with in-context learning. arXiv. https://arxiv.org/abs/2402.12234

[11] Braun, D., Hernandez-Mendez, A., Matthes, F., & Langen, M. (2017). Evaluating natural language understanding services for conversational question answering systems. In Proc. 18th Annu. SIGdial Meeting Discourse Dialogue (pp. 174–185).

[12] Kim, Y., Kim, D., & Kim, G. (2021). Intent detection and slot filling for goal-oriented dialogue systems with BERT-based transfer learning. Information Sciences, 546, 114–125.

[13] Zhang, C., Liu, Q., Huang, M., & Zhu, X. (2020). Multi-turn intent detection via token-level dynamic fusion. In Proc. 58th Annu. Meeting of the Association for Computational Linguistics (ACL 2020) (pp. 5464–5473).

[14] Görgen, L., & Schimmler, M. (2024). Towards dialogue-based, computer-aided software requirements elicitation. arXiv preprint. arXiv:2310.13953.

[15] Kessabi, L., et al. (2024). Large language models for model-driven engineering: A systematic literature review and expert survey. arXiv.https://arxiv.org/abs/2403.10078 (Accessed May 20, 2025).

[16] Zhang, Z., Takanobu, R., Zhu, Q., Huang, M., & Zhu, X. (2020). Recent advances and challenges in task-oriented dialogue systems. In Proc. 58th Ann. Meeting Assoc. Comput. Linguistics (ACL 2020).

[17] Lee, H., & Jeong, O. (2023). A knowledge-grounded task-oriented dialogue system with a hierarchical structure for enhancing knowledge selection. Sensors, 23, 685. https://doi.org/10.3390/s23020685

[18] Dong, Q., et al. (2023). A survey on in-context learning. In Proc. 2023 Conf. Empirical Methods Natural Lang. Process. (EMNLP) (pp. 1107–1128).

[19] Brown, T. B., et al. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems (Vol. 33, pp. 1877–1901).

[20] Xie, S. M., Raghunathan, A., Liu, F., et al. (2022). An explanation of in-context learning as implicit Bayesian inference. In International Conference on Learning Representations (ICLR).

[21] Zhao, J., Gupta, R., Cao, Y., et al. (2022). Description-driven task-oriented dialog modeling. arXiv preprint. arXiv:2201.08904.

[22] Laaz, N., Wakil, K., Gotti, Z., Gotti, S., & Mbarki, S. (2019). Integrating domain ontologies in an MDA based development process of e-health management systems at the CIM level. In Advanced Intelligent Systems for Sustainable Development (AI2SD'2019) Volume 2 (pp. 213–223). Springer International Publishing.

[23] Chen, M., Tworek, T., Jun, H., Yuan, Q., Sorensen, H., Schuster, S., et al. (2021). Evaluating large language models trained on code. arXiv preprint. arXiv:2107.03374.

[24] Kulkarni, V., & Clark, T. (2014). Model driven development of business applications: A practitioner's perspective. Journal of Software: Evolution and Process, 26(6), 613–638.

[25] Kharmoum, N., Ziti, S., Rhazali, Y., & Omary, F. (2019). A method of model transformation in MDA approach from e3value model to BPMN2 diagrams in CIM level. IAENG International Journal of Computer Science, 46(4), 1–17.

[26] Essebaa, I., Chantit, S., & Ramdani, M. (2019). MoDAr-WA: Tool support to automate an MDA approach for MVC web application. Computers, 8(4), 89.

[27] Syvänen, S., Heinonen, K., & Salonen, A. (2020). Conversational agents in online organization–stakeholder interactions: A state-of-the-art analysis and implications for further research. Corporate Communications: An International Journal, 25(4), 613–631.

[28] Melouk, M., Rhazali, Y., & Hadi, Y. (2020). An approach for transforming CIM to PIM up to PSM in MDA. Procedia Computer Science, 170, 869–874.

[29] Kourani, H., Berti, A., Schuster, D., & van der Aalst, W. M. P. (2024). Process modeling with large language models. arXiv. https://arxiv.org/abs/2403.07541