Towards Human-in-the-Loop LLM-Enabled Domain Modeling

Jonathan Silva[1(✉)], Qin Ma[1], Jordi Cabot[1,2], Pierre Kelsen[1], and Henderik A. Proper[3]

[1] University of Luxembourg, Esch-sur-Alzette, Luxembourg
{jonathan.silva,qin.ma,jordi.cabot,pierre.kelsen}@uni.lu
[2] Luxembourg Institute of Science and Technology, Esch-sur-Alzette, Luxembourg
jordi.cabot@list.lu
[3] TU Wien, Vienna, Austria
henderik.proper@tuwien.ac.at

Abstract. The use of Large Language Models (LLMs), combined with advanced prompting strategies, automates the creation of domain models from textual domain descriptions. However, the output is often influenced by mistakes and limitations that arise from the inherent characteristics of LLMs, including hallucinations and inconsistencies. Additionally, ambiguities and incompleteness in the input text further affect the quality of the results.

We propose a new LLM-based modeling method with human in the loop that aims to combine the strengths of automatic model creation with human supervision and interaction to refine and validate the model.

In our approach, the LLM generates an initial draft model from textual descriptions. This draft is then subjected to a feedback loop moderated by a rule-based agent, which engages the user through a Q&A dialogue. The rule-based agent selects the questions based on their potential to clarify the most uncertain aspects of the model up to that point.

Keywords: Domain Modeling · Domain Model Validation · AI-assisted Domain Modeling · Large Language Models

1 Introduction

With the growing interest and adoption of Large Language Models (LLMs) and their rapid update and iteration, the application of LLMs to solve domain modeling problems has also advanced significantly. What began with straightforward single-shot prompting has now evolved into the exploitation of more sophisticated prompting techniques, such as decomposing the modeling process into smaller tasks [4,7,11,30,32,35], assigning specialized roles to LLMs to allow collaborative co-creation [20,22,32], and incorporating self-reflections [17,27,32].

Despite advances, LLM-based domain modeling solutions still face noticeable limitations. For example, in UML class diagram modeling, accurately identifying

D. Bork et al. (Eds.): ER 2025, LNCS 16189, pp. 127–145, 2026.
https://doi.org/10.1007/978-3-032-08623-5_7

relationships among classes remains challenging [11,32,33]. This is largely due to the inherent characteristics of LLMs, including hallucinations and inconsistencies, which can compromise the quality of the output. Moreover, ambiguities and incompleteness in the input text may further degrade the quality of the generated models. As a result, the validity and domain relevance of the created models still require careful verification and validation (V&V).

According to the SEQUAL framework from the model quality V&V literature [21,23], syntactic and semantic quality are considered two fundamental dimensions of model quality. Briefly, syntactic quality refers to a model's adherence to the syntax rules of the modeling language and semantic quality ensures that the model correctly represents the domain and captures all relevant aspects. Recent studies [12] show that when LLMs are used to produce models in widely adopted modeling notations such as (Plant)UML, they tend to produce few syntactic errors, but frequently introduce semantic inaccuracies. The SEQUAL framework recommends consistency checking as a means to achieve semantic validity, whose effective execution requires the involvement of domain experts [19,24], since fully automated validation is considered infeasible [8]. In effect, the specialized knowledge of domain experts is essential to disambiguate and complete the domain understanding captured in textual sources and represented in the model [14,16,18,25].

In line with this, research such as [20] proposed a human-in-the-loop (HITL) approach to LLM-based modeling, aiming to exploit user feedback to refine and eventually enhance the quality of domain models created by LLMs. Specifically, next to leveraging advanced prompting techniques such as role prompting and knowledge injection, [20] incorporated an interactive feedback loop that enables users to identify errors in LLM-generated models and provide corrective feedback, which the LLM then uses to refine the models. This approach demonstrated a promising improvement in the quality of the resulting process models. However, the observed improvement is dependent on users, who are often domain experts, also possessing sufficient modeling expertise. This includes both (1) the ability to comprehend domain models expressed in specialized modeling notations, and (2) the skills to identify and articulate the issues within them. The findings of studies with novice modelers e.g., [33] also support this correlation, indicating that the quality of domain models produced with the assistance of LLMs improves proportionally to the modeling skills and experience of the users.

Given that domain experts often lack a technical modeling background, traditional modeling scenarios, where a domain expert collaborates with a human modeler rather than an LLM, place the responsibility on the modeler to bridge the gap between technical modeling and domain understanding. To enhance model comprehension, the modeler can leverage several strategies such as automatically generating instances from the model and visualizing them using domain-specific notations familiar to the expert [31], transforming the model into natural language representations [8], and explaining the model in natural language through dialogues [5]. These are then followed by strategies to guide and facilitate error detection and feedback elicitation, such as posing pertinent ques-

tions derived from pattern matches in the model [5]. The identified issues and feedback for improvement are then translated by the modeler into appropriate modeling refinements to improve the model [18].

However, when the role of modeler is assumed by an LLM, as is the case in current LLM-based domain modeling solutions, the criticality of bridging the gap between technical modeling and domain understanding is often overlooked. Inspired by the strategies employed by human modelers, this paper proposes an extension to existing LLM-based domain modeling approaches by integrating an interactive feedback loop with the domain expert, moderated by a rule-based agent. Specifically, we demonstrate the extension using a previous LLM-based domain modeling solution built on the Tree-of-Thoughts (ToT) prompting technique [32], which we refer to as the ToT-Q framework. In ToT-Q, an LLM generates an initial draft of a UML class diagram model from a textual description by executing a ToT-based modeling process. The rule-based agent then engages the domain expert in an interactive dialogue, posing questions generated based on pattern matches in the draft model to solicit feedback for refinement.

To support this process, we collect a set of controversial modeling patterns from both the literature [5] and our own modeling experience with UML class diagrams. We then design natural language question templates using domain-specific vocabulary to be triggered when pattern matches are found in the draft model. To prevent overwhelming users with an excessive number of questions, we also put in place a confidence scoring mechanism that selectively involves the user only when the confidence is low or multiple competing modeling options have comparable confidence levels. We validate the ToT-Q framework by conducting experiments using two LLMs with contrasting capabilities (GPT-4o and GPT-4o-mini), which are also employed to simulate the domain expert. Our open-source implementation is available on GitHub for customization of the format of the question templates and the conditions under which questions are triggered.

The rest of the paper is organized as follows. Section 2 reviews the state of the art in LLM-enabled domain modeling and HITL approaches. Section 3 presents the ToT-Q framework for conceptual modeling and elaborates on the technical details of the framework. Section 4 reports the experiments and analyzes the results to evaluate the effectiveness of ToT-Q in improving the general LLM domain modeling capabilities. Section 5 describes the tool support and customization options. Finally, Sect. 6 concludes and points to future directions.

2 State of the Art in LLM-Enabled Domain Modeling

Recent research on creating domain models with LLMs has evolved from using prompting techniques such as Few-shot [9,10,15] or Chain-of-Thought (CoT) prompting [10,27] to dividing the process into smaller tasks. This newer approach uses specialized prompts to identify different model elements [4,7,11,29,30,32,35], which are then combined to create a complete domain model. Approaches based on task decomposition have shown better results over direct prompt strategies [7,29,35]; however, there are still challenges in improving the recognition of

attributes and relationships [11,32]. Furthermore, other approaches assign specialized roles to create process models. For example, one approach uses the role of team leader, process design expert, and process reviewer [22]. Another approach proposes a process owner and a modeling expert to create process models [20]. However, the models still require further refinement to fully align with the process description [20]. While substantial research has focused on improving domain models generated with LLMs, the results show that LLMs have limitations and human feedback is essential to improve the models.

Human in the loop approaches which involve human interaction to guide, supervise, or enhance the learning process [26] play a crucial role to address the limitations. Specifically, Interactive Machine Learning that involves an iterative interaction with human feedback and shared decision-making [13]. Building on these principles, we propose an interactive HITL approach that explicitly structures the collaboration between LLMs and domain experts. Our method extends the ToT framework by iteratively refining the model through a validation loop.

Exploration of HITL approaches has not been widely explored, with only two approaches found in the literature. Norouzifar et al. [28] uses LLMs to generate clarifying questions from process descriptions. The answers serve to create a list of rules that is used to create an initial process model, which is subsequently presented to the domain expert to continue with further refinements of the rules. However, it remains unclear whether LLMs can generate clarifying questions that will resolve the ambiguous parts of the process descriptions. Kourani et al. [20] allowed users to improve the model with rounds of feedback to provide the changes needed to refine the model; then, the LLM refines the model based on this input. Nonetheless, this approach assumes that the user possesses sufficient modeling expertise to understand the process model and continue with the iterative refinement. Moreover, research with novice UML modelers has identified that models created with LLM assistance include errors caused by the known limitations in relationship identification, which are not detected by the users with low modeling expertise [33]. Our approach addresses these limitations through a structured HITL strategy adapted to non-expert users. The rule-based agent generates questions about model elements with low confidence; these questions are created with templates adapted to users with limited UML expertise. This structured interaction ensures that domain experts focus their attention on answering questions about uncertain parts of the model.

3 The TOT-Q Framework

We present an overview of the ToT-Q framework in Fig. 1. ToT-Q structures the modeling process into three main phases, orchestrated by a rule-based agent: 1) the initial phase starts with a domain description that leads to the creation of a draft domain model; 2) the iterative improvement phase refines the domain model through a Q&A loop, and 3) the final phase is reached once all relevant questions are addressed and the improved domain model is proposed to the user. These phases are enabled by four key technical components (cf. Fig. 2).

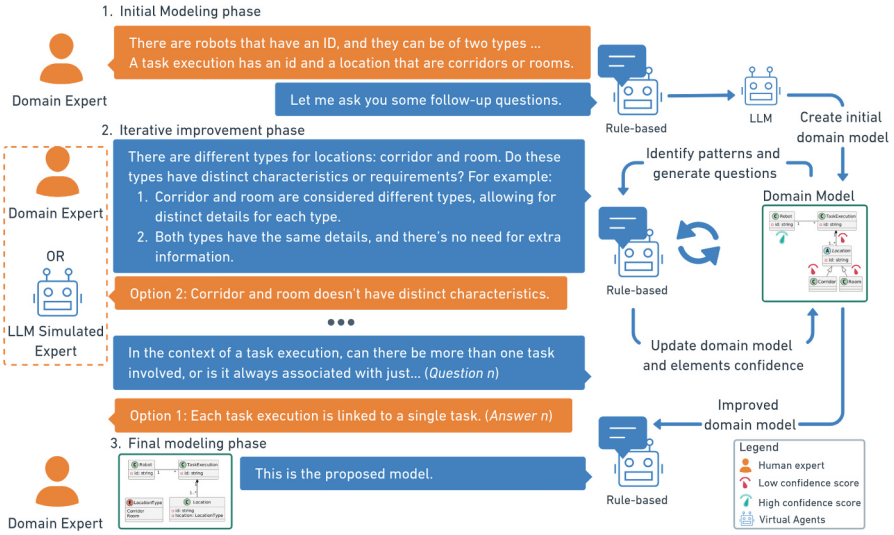


Fig. 1. ToT-Q follows a 3 phases approach moderated by a rule-based agent: 1) Initial modeling phase: delegates the creation of a draft domain model to an LLM, 2) Iterative improvement phase: engages the user in an iterative Q&A loop to refine the domain model, and 3) Final modeling phase: proposes the improved domain model to the user.

1. **ToT and Confidence Quantification:** The initial domain model is created by an LLM using the ToT prompting strategy. During ToT execution, we also let the LLM verbalize its confidence and synthesize it in a hybrid confidence quantification mechanism.
2. **Modeling Pattern Matching:** The rule-based agent identifies matches of modeling patterns in the domain model and prepares data for the generation of questions.
3. **Generation and Selection of Questions:** Questions are generated by instantiating the template linked to the matched modeling pattern. Only those with a confidence score below a defined threshold are presented to the domain expert during the Q&A loop. Moreover, the agent prioritizes these questions by asking those with the lowest confidence scores first.
4. **Model Refinement and Confidence Score Update:** The rule-based agent refines the domain model based on the expert's feedback and updates the corresponding confidence scores, until all questions have been addressed, or a predefined limit on the number of questions has been reached.

We illustrate the ToT-Q framework in the following with an example domain of robots moving in different locations, such as corridors or rooms. Using UML class diagrams, these locations can be either represented as literals of an enumeration type or as subclasses inheriting from a common location class.

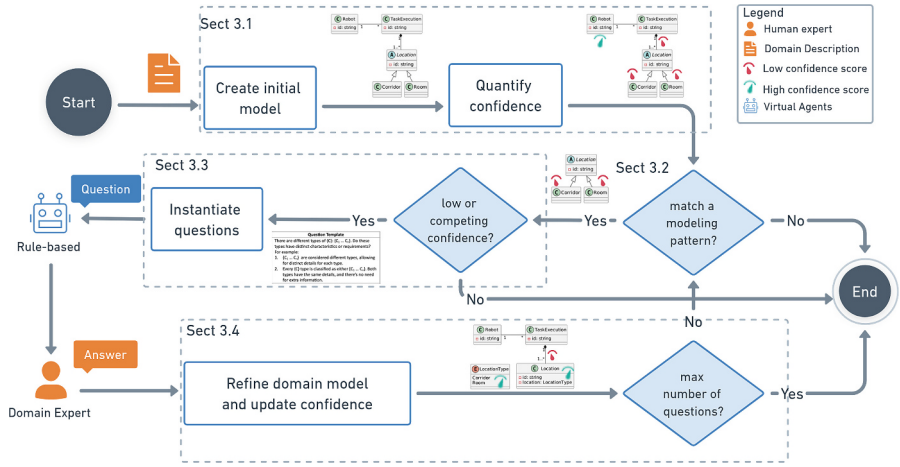


Fig. 2. The four main components of the process refinement process: 1) ToT and confidence quantification, 2) modeling pattern identification, 3) selection and generation of questions, and 4) confidence update based on expert feedback.

3.1 ToT and Confidence Quantification

We use the ToT4DM framework [32] to create the initial domain model. This framework is built upon the ToT prompting strategy [36] designed to enhance the reasoning capabilities of LLMs by decomposing a complex problem into smaller tasks and promoting LLMs to self-assess their responses. In the context of domain modeling, ToT offers the potential to explore and evaluate alternatives of partial models, enabling the creation of domain models with complex abstractions. Specifically, the modeling process is decomposed into a sequence of tasks, referred to as the ToT levels, to build up the classes, attributes, relationships, and association classes of a UML class diagram step by step. For each level, an LLM generates k candidate thoughts, each containing multiple model elements. The LLM then votes for the best thought, and uses the breadth-first search (BFS) to explore the best thought further to complete the draft model. The rest of the thoughts are pruned.

To determine the uncertain parts of the draft model that require domain expert feedback, we quantify it with confidence scores. Our quantification mechanism is inspired by recent advancements in uncertainty quantification for LLMs, particularly methods to estimate confidence in the outputs of closed-source LLMs [37], which do not rely on the logits that are only accessible in open-source LLMs. Specifically, we adopt a hybrid strategy, following the method of Xiong et al. [34], to calculate the confidence scores, combining verbalized confidence scores expressed by LLMs in their responses based on the consistency among the k candidate thoughts of a given level.

To elicit the verbalized confidence scores, we include in the ToT prompts an instruction for the LLM to express a confidence score in $[0, 1]$ alongside each

of the following model elements: classes, attributes, enumerations, relationships, their source and target cardinalities, and association classes.

Given a level, let t_i (for $i = 1, \ldots, k$) range over the k thoughts of this level. A thought t_i contains a set of model elements, each paired with a corresponding verbalized confidence score, namely $t_i = \{(e_1, vc_1), (e_2, vc_2), \ldots\}$. Note that different thoughts within the same level may propose model elements that are in a *semantic match*, meaning the name of the elements refer to the same concept but may be expressed differently [6]. We denote the semantic match relation between two elements as $\approx$.

Briefly, the hybrid confidence score of an element e, denoted by $S(e)$, is defined as the average of the verbalized confidence scores of all elements that are in a semantic match with e across the k thoughts at a given level, namely:

$$S(e) = \frac{1}{k} \sum_{i=1}^{k} vc^i_{\approx e} \tag{1}$$

where $vc^i_{\approx e}$, the verbalized confidence score of an element in a semantic match with e in thought t_i, is defined as:

$$vc^i_{\approx e} = \begin{cases} vc' & \text{if } \exists e'.(e', vc') \in t_i \text{ and } e' \approx e \\ 0 & \text{otherwise} \end{cases}$$

Note that if a thought t_i does not contain an element in a semantic match with e, we assume the LLM implicitly verbalizes a confidence score of 0, reflecting the intuition that absence indicates a lack of confidence.

To illustrate the confidence quantification, consider an example where in the level for class modeling, in two out of the $k = 3$ thoughts, the LLM proposes a corridor class to represent the concept of corridor as a location with verbalized confidence scores of 0.70 and 0.80 respectively. In the remaining thought, the LLM proposes to capture corridor as a literal in an enumeration. The confidence score of the corridor class is thus equal to $(0.7 + 0.8 + 0)/3 = 0.5$, according to Eq. 1.

3.2 Modeling Pattern Matching

In this section, we present the modeling patterns used in our proposal, each of which contains two facets. We then explain how a pattern is matched in the model and how the two facets are paired using Algorithm 1.

Figure 3 defines eight modeling patterns, each containing two facets, representing two alternative options to model a concept in UML class diagrams. The list of patterns was created by combining insights from the patterns identified by Bertolino et al. [5] and our modeling experience with UML class diagrams. The goal is to identify scenarios of uncertainty where the domain concept can be represented in the model with two alternative options, which facilitates the generation of contrastive questions. Determining which option is the most appropriate depends on a detailed analysis of the domain being modeled.

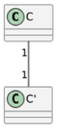
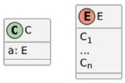
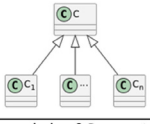
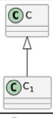
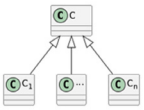
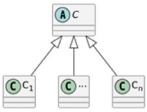
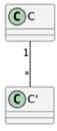
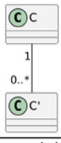
Pattern	Facet 1	Facet 2	Question Template
1	**Decision:** Use a class to represent a concept composed by multiple parts, or an attribute for a simple representation.		
			We know that each {C} has an {C'}. When we think about {C'} is it just a simple piece of information, like a line of text? Or is {C'} made up of different parts that might be important? For example: 1. For each { C'}, the information is saved separately, so users can filter {C} based on {C'} details. 2. The {C'} is a single piece of text. This works well when we just need to show the {C'}.
2	**Decision:** Represent concepts with enumeration literals when values are fixed, or use sub classes when concepts involve different details for each one.		
			There are different types of {C}: {C_1 ... C_n}. Do these types have distinct characteristics or requirements? For example: 1. {C_1 ... C_n} are considered different types, allowing for distinct details for each type. 2. Every {C} type is classified as either {C_1 ... C_n}. Both types have the same details, and there's no need for extra information.
3	**Decision:** Inheritance with one sub class & Represent a concept as an attribute for a simple representation, or a sub class when the concept requires more details.		
			When thinking about {C}, what makes a {C_1} different from a regular one? Is there a specific role or responsibility that comes with being a {C_1}? Are there additional factors we need to consider when dealing with {C_1} compared to regular {C}?. For example: 1. A {C_1} is treated as a different kind of {C}. 2. {C} include a status that identifies if it as {C_1} or not.
4	**Decision:** Represent a concept with a concrete class if it can exists independently as instances; otherwise, use an abstract representation.		
			When we talk about {C}, do we always have a clear idea of a specific example, or do we sometimes speak about it in a more general way? Should we think of {C} as a general concept that covers all possible forms, or is it always tied to a particular type or instance?. For example: 1. A general {C} can exists that is not a {C_1 ... C_n}. 2. A {C} is just a concept and cannot exist on its own. We only need specific types like {C_1 ... C_n}.
5	**Decision:** Use composition when the relationship is strong and has lifecycle dependency; otherwise, use association.		
			Does a {C'} always require a {C} to exist, or could it exist independently and possibly be linked to multiple {C} instances? If a {C} is removed, would all the associated {C'} also need to be removed, or could a {C'} still make sense on its own?. For example: 1. A {C'} belongs to a {C} and cannot exist without a specific {C}. 2. A {C'} is linked to a {C} but can exist independently of it.
6	**Decision:** The lowerbound cardinality is to zero instance, or one instance.		
			In the context of a {C}, does it always involve at least one {C'} or can it exist without any?. For example: 1. A {C} can exist without any {C'} 2. A {C} must have at least one {C'}
7	**Decision:** The upperbound cardinality is to only one instance, or multiple instances.		
			In the context of a {C}, can there be more than one {C'} involved, or is it always associated with just one?. For example: 1. A {C} is linked to a single {C'}. 2. A {C} is linked to multiple {C'}.
8	**Decision:** Represent a concept as an association class when a relationship has attributes; otherwise, use a class.		
			A {C} has multiple {C_{ac}} at different {C'}. Should we think of the link between a {C} and a {C'} as something that only exists when a {C_{ac}} happens? Or would you say a {C} and a {C'} should have a link even outside of specific {C_{ac}}?. For example: 1. A {C} has a direct relationship with a {C'}. {C_{ac}} is a separate event that connects the {C} and {C'} temporarily. 2. The {C_{ac}} is the key event that establishes the link between a {C} and a {C'}. The {C} and {C'} are only linked when a {C_{ac}} occurs.

Fig. 3. Modeling patterns with its corresponding facets, decision, and template question.

Algorithm 1. Find modeling pattern facets

Input: List of facets $F_{\mathcal{DM}}$ with p elements, List of facets $F_{\mathcal{DM}_{alt}}$ with q elements
Output: List containing the alternative pairs: $AlternativePairs$

function FINDALTERNATIVES($F_{\mathcal{DM}}$, $F_{\mathcal{DM}_{alt}}$)
  $AlternativePairs \leftarrow \emptyset$
  for $i \leftarrow 1$ to p **do**
    $\mathcal{A}_{\mathcal{DM}} \leftarrow F_{\mathcal{DM}}[i]$
    $Matched \leftarrow$ **false**
    $j \leftarrow 1$
    while $j \leq q$ and not $Matched$ **do**
      $\mathcal{A}_{\mathcal{DM}_{alt}} \leftarrow F_{\mathcal{DM}_{alt}}[j]$
      if MATCH($\mathcal{A}_{\mathcal{DM}}$, $\mathcal{A}_{\mathcal{DM}_{alt}}$) **then** // True when represent the same concept
        $Matched \leftarrow$ **true**
      end if
      $j \leftarrow j + 1$
    end while
    if not $Matched$ **then**
      $\mathcal{A}_{\mathcal{DM}_{alt}} \leftarrow$ GENERATEALTERNATIVE($\mathcal{A}_{\mathcal{DM}}$) // Transform $\mathcal{A}_{\mathcal{DM}}$ to $\mathcal{A}_{\mathcal{DM}_{alt}}$
    end if
    $AlternativePairs \leftarrow AlternativePairs \cup \{(\mathcal{A}_{\mathcal{DM}}, \mathcal{A}_{\mathcal{DM}_{alt}})\}$
  end for
  return $AlternativePairs$
end function

The draft domain model consists of a set of model elements, denoted by $\mathcal{DM} = \{e_1, e_2, \ldots, e_n\}$. Let $\mathcal{DM}_{alt} = \{e'_1, e'_2, \ldots, e'_m\}$ be the set of model elements of all the pruned thoughts. To match a pattern, we look for a subset of elements $\mathcal{A}_{\mathcal{DM}} \subseteq \mathcal{DM}$ such that $\mathcal{A}_{\mathcal{DM}}$ constitutes an instance satisfying one facet of the pattern, called a match of the facet. If a match of one facet of a pattern is found in $\mathcal{DM}$, we then look for a match of the other facet of the pattern in $\mathcal{DM}_{alt}$, referred to as $\mathcal{A}_{\mathcal{DM}_{alt}} \subseteq \mathcal{DM}_{alt}$, and pair it with $\mathcal{A}_{\mathcal{DM}}$ to constitute a *configuration*. If no match of the other facet can be found in $\mathcal{DM}_{alt}$, we generate $\mathcal{A}_{\mathcal{DM}_{alt}}$ by transforming the elements from $\mathcal{A}_{\mathcal{DM}}$. Algorithm 1 implements this process to iterate through the patterns and generate the configuration pairs:

$$C = \{(\mathcal{A}_{\mathcal{DM}}{}^1, \mathcal{A}_{\mathcal{DM}_{alt}}{}^1), (\mathcal{A}_{\mathcal{DM}}{}^2, \mathcal{A}_{\mathcal{DM}_{alt}}{}^2), \ldots, (\mathcal{A}_{\mathcal{DM}}{}^n, \mathcal{A}_{\mathcal{DM}_{alt}}{}^n)\}$$

where each configuration $(\mathcal{A}_{\mathcal{DM}}{}^i, \mathcal{A}_{\mathcal{DM}_{alt}}{}^i)$, for $i = 1, \ldots, n$, contains instances of the two alternative facets of a modeling pattern from Fig. 3. These configurations provide input data for instantiating the question templates, as elaborated in Sect. 3.3.

As an example, suppose we identify a subset $\mathcal{A}_{\mathcal{DM}}$ of the draft model satisfying Facet 2 of Pattern 2, consisting of a location class and two subclasses: corridor and room, inheriting it. We then look for a match of the other facet of Pattern 2, namely $\mathcal{A}_{\mathcal{DM}_{alt}}$, in the set of model elements that were once proposed by the LLM during ToT execution, but were not retained. If the match cannot be found, $\mathcal{A}_{\mathcal{DM}_{alt}}$ is created by transforming $\mathcal{A}_{\mathcal{DM}}$: (1) the location class is kept;

(2) the two subclasses and the inheritance relationship to the location class are replaced by an enumeration type with two literals, each of which corresponds to one subclass; and (3) a new attribute of the enumeration type is added to the location class. We pair $\mathcal{A}_{\mathcal{DM}}$ with $\mathcal{A}_{\mathcal{DM}_{alt}}$, include $(\mathcal{A}_{\mathcal{DM}}, \mathcal{A}_{\mathcal{DM}_{alt}})$ in the set of configurations C, and continue to match more patterns.

3.3 Generation and Selection of Questions

To select the configurations that require validation with the domain expert, we filter them based on the confidence scores.

Let $C = \{c_1, c_2, \ldots, c_n\}$ be a set of configurations, where each configuration c_i is a pair of alternatives, corresponding to facets of modeling patterns $c_i = ({\mathcal{A}_{\mathcal{DM}}}^i, {\mathcal{A}_{\mathcal{DM}_{alt}}}^i)$.

For a given alternative $\mathcal{A}_{\mathcal{DM}} = \{e_1, e_2, \ldots, e_m\}$, where e_j are the elements that define the distinctive modeling constructs of the facet (e.g. subclasses or an enumeration), the confidence score is $S({\mathcal{A}_{\mathcal{DM}}}^i) = min\{S(e_j)|e_j \in {\mathcal{A}_{\mathcal{DM}}}^i\}$. The same applies to $S({\mathcal{A}_{\mathcal{DM}_{alt}}}^i)$.

For each configuration c_i, we have:

- A confidence score for the alternative in the domain model: $S({\mathcal{A}_{\mathcal{DM}}}^i) \in [0, 1]$.
- A confidence score for the alternative not selected $S({\mathcal{A}_{\mathcal{DM}_{alt}}}^i) \in [0, 1]$.

Given a threshold $T \in [0, 1]$, we define the set of configurations selected for question generation as:

$$C_{\text{selected}} = \left\{ c_i \in C \mid S({\mathcal{A}_{\mathcal{DM}}}^i) \leq T \vee \left(S({\mathcal{A}_{\mathcal{DM}}}^i) \geq T \wedge S({\mathcal{A}_{\mathcal{DM}_{alt}}}^i) \geq T \right) \right\}$$

This selection ensures that configurations with an alternative confidence below the threshold, or those with both alternatives having the confidence above the threshold, are used to trigger questions for expert validation.

The selected configurations are the input to create questions with a heuristic that indicates which modeling option is more appropriate, based on modeling practices or characteristics of the domain. These heuristics are not strict rules; rather, they provide arguments about the two options, helping domain experts compare and select the best one.

Inspired by Cabot et al. [8], who transform UML/OCL models into natural language to aid non-technical stakeholder understanding, and Bertolino et al. [5], who generate natural language text from UML class diagrams, our implementation employs a template-based question generation mechanism. These templates adapt the content to be accessible to experts, without requiring modeling expertise. In Fig. 3, the templates are populated with the model elements in $\mathcal{A}_{\mathcal{DM}}$, ensuring that the questions are based on the domain context.

The purpose is for the rule-based agent to present the questions with contrastive options, where the domain expert selects one of the alternatives presented. This supports the expert in evaluating both modeling alternatives. Then, the answers will guide the improvement of the model based on the decisions made. By combining the patterns described in Sect. 3.2 with a structured set of

template-based questions, we facilitate the participation of domain experts in the modeling process. This enables them to make informed decisions without requiring prior modeling expertise. The complete set of templates and examples is available in GitHub[1].

In the running example, $\mathcal{A}_{\mathcal{DM}}$ contains the subclass representation for corridor and room with a confidence score of 0.50. As this score is below the predefined threshold of 0.80 set in our implementation, the configuration is selected to generate a question for expert validation. Then, the elements class location, and subclasses corridor and room are used to generate the question:

> There are different types of locations: corridors and rooms. Do these types have distinct characteristics or requirements? For example:
> 1. Corridors and rooms are considered different types, allowing for distinct details for each type.
> 2. Every location is classified as either corridor or room. Both types have the same details, and there's no need for extra information.

3.4 Model Refinement and Confidence Score Update

After the domain expert answers the question, one of the alternative pairs $({\mathcal{A}_{\mathcal{DM}}}^i, {\mathcal{A}_{\mathcal{DM}_{alt}}}^i)$ is selected, where $\mathcal{A}_{sel} \in \{{\mathcal{A}_{\mathcal{DM}}}^i, {\mathcal{A}_{\mathcal{DM}_{alt}}}^i\}$ is the selected alternative. $\mathcal{A}_{sel}$ may contain elements that are categorized into two types:

- Trusted elements: Elements whose modeling construct are explicitly confirmed by the answer. For example, if a subclass will be replaced by an enumeration literal, we have a high confidence in the enumeration element in the model.
- Uncertain elements: elements whose modeling construct become uncertain as a result of the answer. For example, when enumeration literals are replaced by a subclass, we have a low confidence whether the parent class should be abstract or concrete.

Detailed examples of trusted and uncertain elements for each modeling pattern used in our approach are provided in the GitHub repository (see footnote 1).

For each model element $e \in \mathcal{A}_{sel}$. We define the following confidence update parameters:

- $S_{\text{low}} \in [0, 0.5]$: minimum confidence for uncertain elements.
- $S_{\text{high}} \in (0.5, 1]$: high confidence for trusted elements.

The confidence update is performed as follows:

$$\forall e \in \mathcal{A}_{sel}, \quad S(e) \leftarrow \begin{cases} S_{\text{high}} & \text{if } e \text{ is a trusted element} \\ S_{\text{low}} & \text{if } e \text{ is an uncertain element} \end{cases}$$

[1] https://github.com/BESSER-PEARL/TOT-Q/blob/main/tot_rules_q/modeling_patterns_templates.pdf.

Using this approach, we update the confidence of the elements involved in the question, assigning a high confidence to the trusted elements, while uncertain elements are assigned a low confidence. By increasing the confidence in model elements, we support the creation of a model that is aligned with the domain expert's knowledge.

In the running example, the expert answer indicates that corridor and room have distinct characteristics. Based on this feedback, the most suitable representation is subclasses. Consequently, the confidence score for the subclass elements is set to $S_{\text{high}} = 0.88$, reflecting high confidence after expert validation. Also, the class location confidence is set to $S_{\text{low}} = 0.44$, due to the uncertainty of whether the class should be represented as abstract or concrete.

4 Evaluation

This section evaluates our approach with the following research questions:

RQ1: How does the inclusion of rule-based generated questions (ToT-Q) impact on precision, recall, and F1 score metrics compared to an LLM only approach?

RQ2: How does the number of elements in the domain model affect the number of questions generated by the ToT-Q iterative improvement?

4.1 Experiments Setup

For the experiments, we execute the ToT only approach with $k = 3$ and compare it with ToT-Q. To fully automate the experiment, we use a second LLM to simulate the domain expert. While this introduces a threat to validity in the results, we believe this is a reasonable trade-off to conduct a more controlled experiment, also given that the domain we target in the experiment is well-known and therefore the LLM can do a good job in answering related questions because it has probably seen it during training. Furthermore, we constrain the LLM's behavior with the system prompt, explicitly instructing it to answer the questions based solely on the textual domain description. It is important to emphasize that we do not claim that LLMs can substitute domain experts. Nevertheless, this setup serves as a controlled evaluation to assess how our approach supports the refinement and improvement of a domain model compared to a direct approach where no experts are involved. If there were human experts, the improvement would be even better, especially for complex domains, as the human would perform better than the LLM playing the role of expert in this simulation.

To answer the research questions, we use two OpenAI LLMs: 1) GPT-4o: this model is the most capable outside the o-series models [1], and 2) GPT-4o-mini: the smaller model that provides similar results to GPT-4o on certain tasks [2]. To address RQ1, we compare the improvement on the performance metrics using GPT-4o to GPT-4o-mini. Then, to answer RQ2, we compare the number of questions triggered by ToT-Q using the two LLM models.

Table 1. Performance Comparison of ToT and ToT-Q using GPT-4o and GPT-4o-mini per model elements

Model Element	ToT			ToT-Q			ToT vs ToT-Q	
	Precision	Recall	F1-score	Precision	Recall	F1-score	Δ F1-score	Δ (%)
GPT-4o								
Class	0.800	0.914	0.853	0.943	0.943	0.943	+ 0.090	+ 10%
Attributes	0.586	0.850	0.694	0.773	0.850	0.810	+ 0.116	+ 17%
Relationships	0.400	0.583	0.475	0.692	0.750	0.720	+ 0.245	+ 52%
Assoc. Class	0.600	0.667	0.632	1.000	0.667	0.800	+ 0.168	+ 27%
GPT-4o-mini								
Class	0.775	0.886	0.827	0.914	0.914	0.914	+ 0.088	+ 11%
Attributes	0.452	0.825	0.584	0.593	0.875	0.707	+ 0.123	+ 21%
Relationships	0.353	0.500	0.414	0.452	0.583	0.509	+ 0.095	+ 23%
Assoc. Class	0.500	0.333	0.400	1.000	0.444	0.615	+ 0.215	+ 54%

The dataset includes five domain descriptions with the reference solution from [32], which includes experiments from previous research with LLMs [12], exercises from a university modeling course, and cases proposed by us. The total number of elements includes: 35 classes, 40 attributes, 24 relationships, and 9 association classes.

4.2 Evaluation Criteria

We manually evaluate the models generated by ToT and ToT-Q by doing a comparison against the reference solution in PlantUML. A model element is considered correct if it is in a semantic match with the corresponding element in the reference model, as defined in Sect. 3.1. Furthermore, we are not evaluating the type of the attributes, the relationships should match the lower and upper cardinality, and we consider zero as the default lower cardinality if not specified.

We use precision, recall, and the F1 score to evaluate the experiments. We consider the correct predictions of model elements as true positives, incorrect predictions as false positives, and the elements not predicted as false negatives.

4.3 RQ1: Impact of Using ToT-Q Compared to LLM Only

Our experiment with ToT-Q and GPT-4o in Table 1 shows performance improvements for all the model elements. For classes, both precision and recall increase, resulting in an F1-score of 0.943, an improvement of 10%. Similarly, for relationships, the performance increases for all metrics, with an F1-score of 0.720, an improvement of 52%. Regarding attributes, precision increases while recall remains the same, leading to an F1-score of 0.810, an increase of 17%. Finally, for association classes the recall is the same while precision increases, resulting in an F1-score of 0.800, with an increase in performance of 27%.

In Table 1, the comparison between ToT using GPT-4o and GPT-4o-mini, the smaller LLM model, shows a decrease in performance across all model element types with GPT-4o-mini underperforming relative to GPT-4o. Regarding the experiments using ToT-Q and GPT-4o-mini approach, we observe improvements in the precision, recall, and F1-score for all model elements compared with the ToT. Specifically, the F1-score increases by 11% for classes, 21% for attributes, 23% for relationships, and 54% in association classes. These findings suggest that even smaller models with restricted performance in modeling tasks can benefit from the ToT-Q approach to improve the model recommendations when no human expert is available. The structured Q&A refinement loop can help to compensate for the limitations of smaller LLMs when generating UML models.

These results indicate that ToT-Q achieves improvements ranging from 10% to 54% depending on the element type. Furthermore, using ToT-Q with GPT-4o, we overcome the limitation of relying on LLM only solutions, which are limited by the capabilities of the underlying LLM. The ToT-Q approach shows that incorporating questions discovered with rules and modeling patterns is a viable strategy to initiate a structured refinement loop. These questions emphasize parts of the model that have low confidence and require feedback from domain experts to have a more accurate representation of the domain.

Table 2. Summary of number of elements, number of questions generated, and ratio of questions generated per element (Q/E) across five domains

Domain	Elements	GPT-4o		GPT-4o-mini	
		Questions	Q/E	Questions	Q/E
E-commerce	24	15	0.63	15	0.63
Person & Resources	22	17	0.77	22	1.00
Theaters	25	11	0.44	6	0.24
Courses	20	25	1.25	22	1.10
Robots	17	4	0.24	11	0.65
Total	**108**	**72**	**0.67**	**76**	**0.70**

4.4 RQ2: Impact of Model Elements in Number of Questions

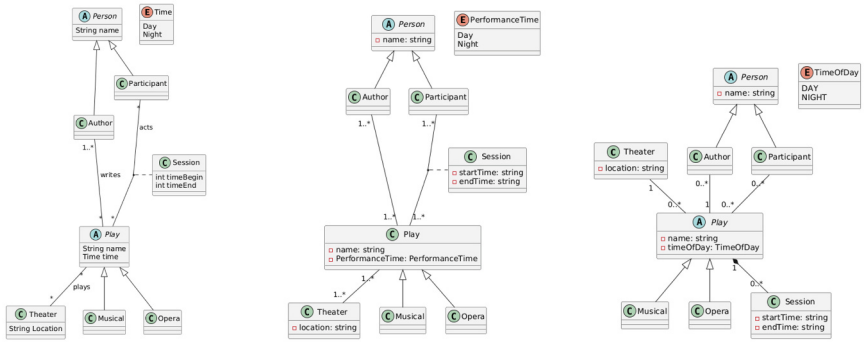
In Table 2, for each domain, we summarize the model size by counting the total number of elements, the number of questions asked by ToT-Q, and the ratio of questions generated by the number of elements. We observe that the smallest model, Robots, with 17 elements, tends to generate fewer questions compared to the larger models, with 4 questions for GPT-4o and 11 for GPT-4o-mini. However, larger models with between 20 and 25 elements show a more complex behavior, as they do not consistently generate more questions. For instance, using ToT-Q with GPT-4o, the Theaters domain with 25 elements requires 11

questions, whereas Courses with 20 elements requires 25 questions. This finding suggests that the number of questions is also influenced by other factors, such as variation in details and ambiguity within the textual descriptions of these domains, which in turn affect the number of questions generated.

Furthermore, we notice that GPT-4o-mini generates more questions overall (76), compared to GPT-4o (72); however, this behavior is not consistent across all the domains. The overall ratio of questions per element is very similar between both LLM models, 0.67 for GPT-4o and 0.7 for GPT-4o-mini, suggesting that on average, about two-thirds of the elements trigger a question in each case.

However, fewer questions do not always result in better models. As shown in Fig. 4, the model generated by ToT-Q with GPT-4o-mini for the Theaters domain differs from the reference solution in three important elements: 1) fails to recognize the association between session and participant, 2) session is not modeled as an association class, and 3) does not reflect in the cardinality that plays have one or more authors.

These results suggest that the size of the model is not the only factor that influences the number of questions. Other factors required may include the ambiguity and completeness of the textual description, as well as the elements that are discovered during ToT execution, influenced by the LLM model. Importantly, despite these variations, our experiments show that the total number of questions remains reasonable. This indicates that the ToT-Q interaction process can guide the model refinement with a manageable number of questions.



(a) Reference solution (b) ToT-Q (GPT-4o) (c) ToT-Q (GPT-4o-mini)

Fig. 4. Theater domain model used in the experiments

4.5 Threats to Validity

Simulated LLM Domain Expert. LLMs are generative models that produce different responses when given identical prompts. We reduce this variability by using a system prompt to reply using only the domain description, and changing the temperature of the LLM model to zero. While a temperature of zero reduces response variability[2], it does not guarantee complete determinism.

Confidence Variability Using ToT. The ToT approach can lead to variability in confidence levels associated with the model elements. To address this, we employ a hybrid approach that incorporates self-consistent confidence. Using this method, we consider the frequency of the model elements suggested across the ToT multiple paths in the confidence quantification.

Limited Number of Reference Models. Our dataset covers reference models for five domains, which may not represent the complexity and ambiguity that exists in real scenarios. This limitation could affect the generalization of our findings, highlighting the need for evaluation with more diverse and larger models.

5 Tool Support

Our open-source tool is available on GitHub[3]. For the ToT implementation, we use the tool from our previous work [32] to create the initial domain model, which uses the OpenAI API or Azure OpenAI API. We implement the rule-based agent that asks questions to the domain expert; using the BESSER Agentic framework[4], which is part of the BESSER low-code platform [3]. Users can customize the threshold to limit the number of questions, as well as the high and low confidence values by modifying the configuration file[5], which also includes recommended settings. Furthermore, the text for the questions templates can be customized in the file[6].

6 Conclusion and Futher Work

Using LLM only approaches to create domain models often results in incomplete or inaccurate representations. To address this limitation, we propose ToT-Q, a HITL approach that targets uncertain parts of the model that require knowledge from domain experts. Rather than performing an exhaustive revision of all the elements, our method employs a rule-based agent to generate contrastive questions, helping experts to iteratively evaluate and improve the model. Our

[2] https://platform.openai.com/docs/api-reference/chat.
[3] https://github.com/BESSER-PEARL/TOT-Q.
[4] https://github.com/BESSER-PEARL/BESSER-Agentic-Framework/.
[5] https://github.com/BESSER-PEARL/TOT-Q/blob/main/.env_example.
[6] https://github.com/BESSER-PEARL/TOT-Q/blob/main/tot_rules_q/template_questions.py.

open-source tool supports customization of question templates, the number of questions triggered, and the update of confidence score values. Furthermore, we envision extending our proposal by incorporating ontologies and complementary AI techniques to improve element discovery and confidence estimation.

We recognize that human experts have varying expertise levels, which requires adaptability to their diverse profiles. Future work will explore including more modeling patterns and adapting the technical level of the generated questions to different user profiles. Additionally, incorporating a mechanism to capture the degree of certainty in domain expert answers and to aggregate feedback from multiple experts to increase the overall confidence in the model.

While our evaluation relied on simulated expert feedback, it does not fully capture the contextual understanding of human domain experts. Therefore, further work is required to evaluate the effectiveness of the generated questions with domain experts and the quality of models in real scenarios. We expect that involving human experts will lead to even greater improvements.

References

1. GPT-4o (2024). https://platform.openai.com/docs/models/gpt-4o. Accessed 19 May 2025
2. GPT-4o mini (2024). https://platform.openai.com/docs/models/gpt-4o-mini. Accessed 19 May 2025
3. Alfonso, I., et al.: Building BESSER: an open-source low-code platform. In: International Conference on Business Process Modeling, Development and Support, pp. 203–212. Springer (2024)
4. Arulmohan, S., Meurs, M.J., Mosser, S.: Extracting domain models from textual requirements in the era of large language models. In: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 580–587. IEEE (2023)
5. Bertolino, A., De Angelis, G., Di Sandro, A., Sabetta, A.: Is my model right? Let me ask the expert. J. Syst. Softw. **84**(7), 1089–1099 (2011). https://doi.org/10.1016/j.jss.2011.01.054. https://www.sciencedirect.com/science/article/pii/S0164121211000446
6. Bian, W., Alam, O., Kienzle, J.: Automated grading of class diagrams. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), pp. 700–709. IEEE (2019)
7. Bragilovski, M., Can, A.T., Dalpiaz, F., Sturm, A.: Leveraging machines to derive domain models from user stories. Requirements Eng. (2025). https://doi.org/10.1007/s00766-025-00442-9
8. Cabot, J., Pau, R., Raventós, R.: From UML/OCL to SBVR specifications: a challenging transformation. Inf. Syst. **35**(4), 417–440 (2010). https://doi.org/10.1016/j.is.2008.12.002. https://www.sciencedirect.com/science/article/pii/S030643790800094X
9. Chaaben, M.B., Burgueño, L., Sahraoui, H.: Towards using few-shot prompt learning for automating model completion. In: 2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), Melbourne, Australia, pp. 7–12. IEEE (2023). https://doi.org/10.1109/ICSE-NIER58687.2023.00008. https://ieeexplore.ieee.org/document/10173909/

10. Chen, K., Yang, Y., Chen, B., Hernández López, J.A., Mussbacher, G., Varró, D.: Automated domain modeling with large language models: a comparative study. In: 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), Västerås, Sweden, pp. 162–172. IEEE (2023). https://doi.org/10.1109/MODELS58315.2023.00037. https://ieeexplore.ieee.org/document/10344012/
11. Chen, R., Shen, J., He, X.: A model is not built by a single prompt: LLM-based domain modeling with question decomposition. arXiv preprint arXiv:2410.09854 (2024)
12. Cámara, J., Troya, J., Burgueño, L., Vallecillo, A.: On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. Softw. Syst. Model. **22**(3), 781–793 (2023). https://doi.org/10.1007/s10270-023-01105-5
13. Fails, J.A., Olsen Jr, D.R.: Interactive machine learning. In: Proceedings of the 8th International Conference on Intelligent User Interfaces, pp. 39–45 (2003)
14. Feltus, C., Ma, Q., Proper, H.A., Kelsen, P.: Towards AI assisted domain modeling. In: Reinhartz-Berger, I., Sadiq, S. (eds.) ER 2021. LNCS, vol. 13012, pp. 75–89. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88358-4_7
15. Fill, H.G., Fettke, P., Köpke, J.: Conceptual modeling and large language models: impressions from first experiments with ChatGPT. Enterprise Modelling and Information Systems Architectures (EMISAJ), pp. 3:1–15 Pages (2023). https://doi.org/10.18417/EMISA.18.3. https://emisa-journal.org/emisa/article/view/318
16. Frederiks, P.J.M., Weide, T.P.: Information modeling: the process and the required competencies of its participants. Data Knowl. Eng. **58**(1), 4–20 (2006). https://doi.org/10.1016/j.datak.2005.05.007
17. Görgen, L., Müller, E., Triller, M., Nast, B., Sandkuhl, K.: Large language models in enterprise modeling: case study and experiences. In: Mayo, F.J.D., Pires, L.F., Seidewitz, E. (eds.) Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering, MODELSWARD 2024, Rome, Italy, 21–23 February 2024, pp. 74–85. SCITEPRESS (2024)
18. Hoppenbrouwers, S.J.B.A., Proper, H.A.E., Weide, T.P.: A fundamental view on the process of conceptual modeling. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 128–143. Springer, Heidelberg (2005). https://doi.org/10.1007/11568322_9
19. Hoppenbrouwers, S.J., Proper, H., van der Weide, T.P.: Formal modelling as a grounded conversation, pp. 139–155 (2005)
20. Kourani, H., Berti, A., Schuster, D., Aalst, W.M.P.: Process modeling with large language models. In: van der Aa, H., Bork, D., Schmidt, R., Sturm, A. (eds.) Enterprise, Business-Process and Information Systems Modeling, pp. 229–244. Springer, Cham (2024)
21. Krogstie, J., Sindre, G., Lindland, O.I.: 20 Years of Quality of Models, pp. 103–107. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36926-1_8
22. Lin, L., Jin, Y., Zhou, Y., Chen, W., Qian, C.: Mao: a framework for process model generation with multi-agent orchestration arXiv:2408.01916 (2024). https://doi.org/10.48550/arXiv.2408.01916. arXiv:2408.01916
23. Lindland, O., Sindre, G., Solvberg, A.: Understanding quality in conceptual modeling. IEEE Softw. **11**(2), 42–49 (1994). https://doi.org/10.1109/52.268955
24. Ma, Q., Kaczmarek-Heß, M., Kinderen, S.: Validation and verification in domain-specific modeling method engineering: an integrated life-cycle view. Softw. Syst. Model. **22**(2), 647–666 (2023). https://doi.org/10.1007/s10270-022-01056-3

25. Martini, M., Pinggera, J., Neurauter, M., Sachse, P., Furtner, M.R., Weber, B.: The impact of working memory and the "process of process modelling" on model quality: investigating experienced versus inexperienced modellers. Sci. Rep. **6**(1), 25561 (2016)
26. Mosqueira-Rey, E., Hernández-Pereira, E., Alonso-Ríos, D., Bobes-Bascarán, J., Fernández-Leal, Á.: Human-in-the-loop machine learning: a state of the art. Artif. Intell. Rev. **56**(4), 3005–3054 (2023). https://doi.org/10.1007/s10462-022-10246-w
27. Neuberger, J., Ackermann, L., van der Aa, H., Jablonski, S.: A universal prompting strategy for extracting process model information from natural language text using large language models. In: Maass, W., Han, H., Yasar, H., Multari, N.J. (eds.) Conceptual Modeling - 43rd International Conference, ER 2024, Pittsburgh, PA, USA, 28–31 October 2024, Proceedings. Lecture Notes in Computer Science, vol. 15238, pp. 38–55. Springer (2024). https://doi.org/10.1007/978-3-031-75872-0_3
28. Norouzifar, A., Kourani, H., Dees, M., van der Aalst, W.M.: Bridging domain knowledge and process discovery using large language models. In: International Conference on Business Process Management, pp. 44–56. Springer (2024)
29. Nour Eldin, A., Assy, N., Anesini, O., Dalmas, B., Gaaloul, W.: A decomposed hybrid approach to business process modeling with LLMs. In: Comuzzi, M., Grigori, D., Sellami, M., Zhou, Z. (eds.) Cooperative Information Systems, vol. 15506, pp. 243–260. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-81375-7_14
30. Prokop, D., Stenchlák, Š., Škoda, P., Klímek, J., Nečaskỳ, M.: Enhancing domain modeling with pre-trained large language models: an automated assistant for domain modelers. In: International Conference on Conceptual Modeling, pp. 235–253. Springer (2024)
31. Razo-Zapata, I., Chew, E., Ma, Q., Gammaitoni, L., Proper, H.: Enabling value co-creation in customer journeys with viva. In: Joint International Conference of Service Science and Innovation and Serviceology (2018)
32. Silva, J., Ma, Q., Cabot, J., Kelsen, P., Proper, H.A.: Application of the tree-of-thoughts framework to LLM-enabled domain modeling. Lecture Notes in Computer Science, vol. 15238, p. 94–111. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-75872-0_6
33. Wang, B., Wang, C., Liang, P., Li, B., Zeng, C.: How LLMs aid in UML modeling: an exploratory study with novice analysts. In: 2024 IEEE International Conference on Software Services Engineering (SSE), pp. 249–257. IEEE (2024)
34. Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., Hooi, B.: Can LLMs express their uncertainty? An empirical evaluation of confidence elicitation in LLMs arXiv:2306.13063 (2024). arXiv:2306.13063
35. Yang, Y., Chen, B., Chen, K., Mussbacher, G., Varró, D.: Multi-step iterative automated domain modeling with large language models. In: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, pp. 587–595 (2024)
36. Yao, S., et al.: Tree of thoughts: deliberate problem solving with large language models. In: Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., Levine, S. (eds.) Advances in Neural Information Processing Systems, vol. 36, pp. 11809–11822. Curran Associates, Inc. (2023). https://proceedings.neurips.cc/paper_files/paper/2023/file/271db9922b8d1f4dd7aaef84ed5ac703-Paper-Conference.pdf
37. Zhao, H., et al.: Explainability for large language models: a survey. ACM Trans. Intell. Syst. Technol. **15**(2), 1–38 (2024)