Leveraging LLMs for Domain Modeling: The Impact of Granularity and Strategy on Quality

Iris Reinhartz-Berger[1][0000−0002−1419−4905], Syed Juned Ali[2][0000−0002−0710−8052], and Dominik Bork[2][0000−0001−8259−2297]

[1] University of Haifa, Haifa, Israel
`iris@is.haifa.ac.il`
[2] TU Wien, Business Informatics Group, Vienna, Austria
`{syed.juned.ali, dominik.bork}@tuwien.ac.at`

Abstract. The information systems engineering community is increasingly exploring the use of Large Language Models (LLMs) for a variety of tasks, including domain modeling, business process modeling, software modeling, and systems modeling. However, most existing research remains exploratory and lacks a systematic approach to analyzing the impact of prompt content on model quality. This paper seeks to fill this gap by investigating how different levels of description *granularity* (whole text vs. paragraph-by-paragraph) and modeling *strategies* (model-based vs. list-based) affect the quality of LLM-generated domain models. Specifically, we conducted an experiment with two state-of-the-art LLMs (GPT-4o and Llama-3.1-70b-versatile) on tasks involving use case and class modeling. Our results reveal challenges that extend beyond the chosen granularity, strategy, and LLM, emphasizing the importance of human modelers not only in crafting effective prompts but also in identifying and addressing critical aspects of LLM-generated models that require refinement and correction.

Keywords: Domain modeling · Conceptual modeling · LLM · Generative AI · UML.

1 Introduction

Domain modeling plays a central role in designing and developing complex systems that address diverse human, organizational, and technological needs. Over time, the discipline has witnessed significant advancements in supporting or automating various modeling tasks. Among these, Large Language Models (LLMs) have recently emerged as powerful tools for natural language understanding and generation, offering promising applications in domain modeling [8,19]. By leveraging their ability to process and analyze large volumes of textual data, LLMs hold the potential to assist in generating accurate and comprehensive models.

Effective modeling requires not only domain knowledge but also methodological rigor. In our previous study [1], we demonstrated that novice modelers (namely, IS students) tend to adopt one of two primary strategies: starting from lists of model elements (e.g., classes) or directly constructing models. Additionally, they often rely on complete textual descriptions rather than incrementally building models from individual paragraphs. These tendencies raise critical questions about how different modeling strategies and granularity levels influence the quality of the resulting models. In this context, model quality is assessed in terms of semantic properties, particularly *requirements satisfaction* and *redundancy*, and syntactic properties, particularly *syntactic correctness* [15,14].

To explore the impact of granularity and modeling strategy on the quality of the LLM-generated domain model, we used two state-of-the-art LLMs, GPT-4o (GPT for short) and Llama-3.1-70b-versatile (Llama for short), to assist with modeling tasks across three distinct application domains. The research is guided by the following research questions:

[RQ1] To what extent does the description ***granularity*** (*whole text* vs. *paragraph-by-paragraph*) affect the quality of LLM-generated domain models?

[RQ2] To what extent do different modeling ***strategies*** (*model-based* vs. *list-based*) influence the quality of LLM-generated domain models?

[RQ3] Are there differences in the observations noted in RQ1 and RQ2 across
- **[RQ3.1]** different ***LLMs*** (*GPT* and *Llama*)?
- **[RQ3.2]** different ***application domains***?
- **[RQ3.3]** different ***tasks*** (Use Case Modeling and Class Modeling)?

The remainder of this paper is structured as follows. Section 2 provides a brief overview of related work on domain modeling and LLMs. Section 3 describes the research methodology, including the experimental design and evaluation metrics. Section 4 presents the results and discusses their implications as well as threats to validity. Finally, Section 5 concludes the paper by summarizing key findings and outlining directions for future research.

2 Related Work

Next, we review key works related to automated domain modeling in general and specifically using LLMs. We also discuss prompting techniques in this context.

2.1 Automated Domain Modeling

Automated domain modeling has been an active research area even before the emergence of LLMs. Traditionally, these methods use textual descriptions of domains and apply statistical or rule-based methods combined with natural language processing (NLP). Such methods have been used to directly derive complete domain models [10,18,17,24], or provide modeling assistance or recommendations during the modeling process [4,20]. Burgueno et al. [4], for example, use vector representations, i.e., word embeddings, to capture the lexical and semantic information from textual documents to suggest model elements for a given partially completed model. Several other approaches combine NLP and machine learning techniques to automate the model creation process [18,24].

Rule-based methods use manually designed grammatical templates to extract domain models from textual descriptions. Robeer et al. [17], for example, present an algorithm with 23 heuristics to automatically identify model elements from user stories. Herchi et al. [11] combine NLP techniques like sentence splitting, tokenization, and syntactic parsing to decompose the input text and then use linguistic rules (e.g., all nouns are converted to entity types) to extract UML concepts. Jahan et al. [12] present a rule-based approach for automated domain modeling and report that their approach effectively produces relevant, simplified diagrams for straightforward user stories, whereas the LLM tends to create more complex diagrams that can go beyond the simplicity of the original user stories.

2.2 LLM-based Domain Modeling

With the advancement of LLMs across various tasks, their application in domain modeling has gained significant attention. Table 1 compares studies in this area, emphasizing the granularity they address, whether they explicitly support refinement or updates to outcomes from prior interactions, the prompt content, and the expected modeling artifacts. Prompt content may encompass task descriptions, domain descriptions, and/or format specifications for the syntax of the expected output.

Fill et al. [9] conducted experiments with GPT-4 for creating ER models, UML class diagrams, and BPMN models. They concluded that very large model parts can be correctly generated by ChatGPT. However, modeling experience is still required to validate the results. Bajaj et al. [3] concluded that GPT-3 outperforms classical tools commonly used in practice for extracting use cases. Chen et al. [8] present a comparative analysis of GPT-4 and GPT-3.5 for automated domain modeling, utilizing various prompt engineering techniques on a dataset comprising ten diverse domain modeling examples. Each example was accompanied by a reference solution created by modeling experts. The authors use the task, domain, and expected output format descriptions to create class diagrams. They conclude, that, while the LLMs demonstrate impressive domain understanding capabilities, they are still impractical for fully automated domain modeling. They further report that LLMs offer the lowest performance in identifying relationships compared to their performance with classes and attributes.

In another work, Chen et al. [6] explore the use of GPT-4 for creating goal models and report that the amount of domain information in the textual description has a limited effect on the responses of GPT-4. The responses have to be evaluated carefully as many elements generated by GPT-4 may be either incorrect or rather generic. Further, the authors conclude that immediate interactive feedback can improve the syntax and semantics of the goal model and expand the initial draft for simple requests. In [7], the authors compare the effectiveness of prompt engineering and fine-tuning for domain-specific modeling tasks. Their findings reveal that approaches focusing on prompt improvements outperform fine-tuning-based methods, even without explicit training on the dataset. Furthermore, the performance gap between prompting and fine-tuning becomes wider when the training dataset is small.

Table 1: Comparison of existing studies in LLM-based domain modeling

Paper	**Granularity**	**Refinement**	**Prompt Content**	**Modeling Artifact**
[9]	Whole Text	Not Supported	Task, Domain, Format	Class Diagrams, ER Diagrams, BPMN Models
[3]	Whole Text	Not Supported	Task, Domain	Use Case Diagrams
[8]	Whole Text	Not Supported	Task, Domain, Format	Class diagrams
[6]	Whole Text	Partially	Task, Domain, Format	Goal Models
[7]	Concept Names	Not Supported	Task, Concept Names	Taxonomy Graphs
[2]	Single User, Story Paragraph	Not Supported	Task, User Story	Class Diagrams
[5]	Whole Text, Modeling Element	Not Supported	Task, Domain	Class diagrams
[19]	Whole Text	Not Supported	Task, Domain	Class Diagrams
Ours	Whole Text, Paragraphs	Supported	Task, Domain, Format	Use Case Diagrams, Class Diagrams

Arulmohan et al. [2] use GPT-3.5 to extract domain models from user stories. The authors apply OpenAI's prompt engineering techniques[1] to create the prompts for the LLM. They separately extract the concepts and relationships and report that while LLMs demonstrate impressive performance, traditional NLP techniques outperform them in this task. Camara et al. [5] also assess GPT-3.5 in an interactive mode using prompts enriched with OCL constraints. They report that GPT-3.5 struggles to handle models larger than 8-10 classes, but performs well with syntax. However, it faces challenges with model semantics. Most notably, the authors report that multiple iterations with explicit requests for modification are required to align the model with the user's intent. Thus, developing a model usually involves an ongoing dialogue with ChatGPT rather than a simple request-response interaction.

In a recent work, Silva et al. [19] proposed a framework to break the task of model generation into separated tasks of class, attribute, and relationship generation using a tree-of-thought framework that allows an LLM to explore several possibilities in the solution space and then choose the best alternative. Their approach performs well to accurately predict the classes, but, similar to the findings in [8], struggles with relationships. Moreover, they do not provide a comparative analysis with existing works rendering the generalizability of their approach unclear.

2.3 Prompting Techniques for Domain Modeling

White et al. [21,22] provide a catalog of prompt engineering techniques that have been applied to solve common problems when interacting with LLMs. The authors conclude that these prompt patterns significantly enrich the capabilities that can be achieved in a conversational LLM. Furthermore, they conclude that

[1] `https://platform.openai.com/docs/guides/gpt-best-practices`

prompt patterns are generalizable to many different domains. Kim et al. [13] formulate prompt templates and conduct comprehensive experiments to assess the impact of in-context examples on LLM-based evaluation. Their experiments reveal that providing clear and straightforward instructions akin to those explained to humans proved to be more effective compared to unstructured and unclear prompts.

2.4 Synopsis

Based on the analysis of existing related studies, we identify the following key observations. First, the quality of the generated domain models is largely influenced by the prompt. Iterative improvements to the initial results are always required but are not explicitly supported in existing works. Second, most studies do not consider the granularity of the domain description. Typically, the entire text is used to generate the models without breaking it into smaller pieces that can fit better within the content window and be incrementally used to build the final domain model. Some works, e.g., [2], do break the task into sub-tasks of generating the classes and relationships separately. However, they do so with the entire domain description. Moreover, they do not combine the domain models generated from each (story) paragraph to produce a consolidated model.

To address these gaps, this study focuses on the granularity of domain descriptions used in input prompts and the strategies employed for model generation. Accordingly, explicit support for model improvements through update operations is incorporated into the incremental generation process.

3 Research Methodology

3.1 Experimental Design and Scenarios

Fig. 1 provides a high-level overview of the experimental design, which involves eight scenarios, generated from three binary variables: granularity (whole text vs. by-paragraph), modeling strategy (model-based vs. list-based), and task. In the context of granularity, "whole text" refers to the complete domain description provided in its entirety, without any segmentation or breakdown. This is the full context given to the model in one go. "By-paragraph" means the domain description is split into individual paragraphs, and each paragraph is treated as a separate input to the model. The idea is to process and analyze each part sequentially or independently. Each paragraph typically focuses on a specific sub-topic or aspect of the domain. In the case of strategy, while we note that different techniques or instructions can be provided to the LLM about *how* to construct a model or list, in this work, we utilize the LLMs *understanding* or *capability* of constructing a list or a model from the domain description by providing straightforward instructions as shown in Table 3. We utilized two common domain modeling tasks: functional modeling via use case diagrams (UCD) and structural modeling via class diagrams (CD). The eight scenarios are summarized in Table 2, while Fig. 2 shows the required interactions with the LLMs

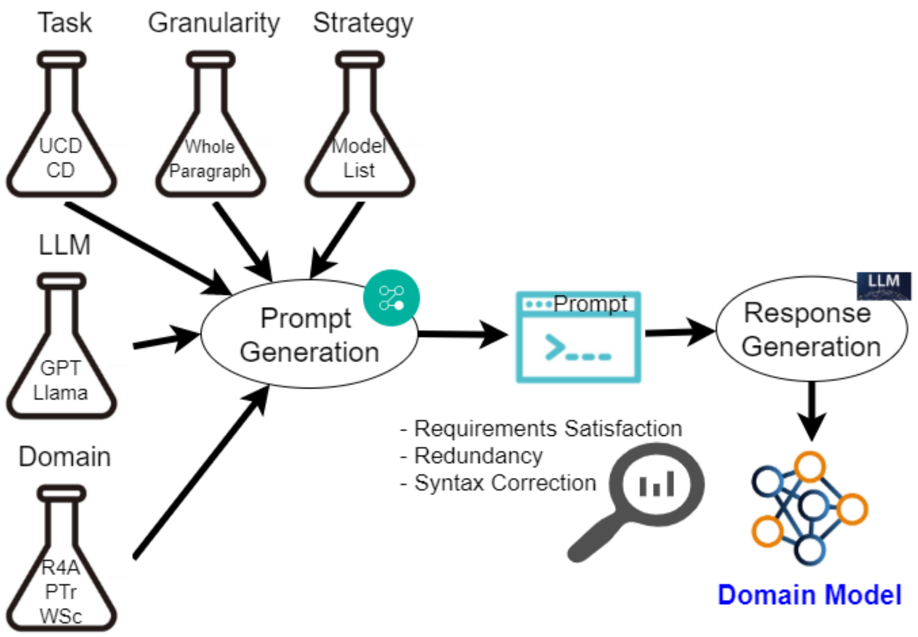

Fig. 1: High-level overview of the experimental design

for each scenario. For scenarios 5 to 8, we instructed the LLMs to first generate separate lists of actors and use cases for use case modeling, and lists of classes for class modeling. These lists were subsequently used as the basis for model generation.

The scenarios were applied to three distinct application domains using two LLMs: GPT-4o and Llama-3.1-70b-versatile. The investigated application domains were: R4A (Rating for All) for viewership data analytics, PTr (Perfect Trip) for tourism management, and WSc (Witchery School) for school acceptance management. To avoid introducing additional confounding variables, the domains were kept to a comparable size and described using seven paragraphs of text each, referring to different aspects of the domains. The entire experimental material can be found in [16].

3.2 Templates and Tasks

Building on our findings in [1], we employed a set of five templates designed to support the key tasks of domain modeling: *Create List* and *Update List* were used to respectively create and incrementally update a list of elements, such as use cases, actors, or classes; *Generate Model* was used to generate a model (e.g., use case or class diagram) from the previously created and potentially updated lists; and *Create Model* and *Update Model* facilitated the direct creation and incremental refinement of models. In all scenarios, we requested that the models

Table 2: The experimental scenarios

Scenario	Granularity	Strategy	Task	# of prompts
1	Whole	Model	UCD	1
2	Whole	Model	CD	1
3	By-paragraph	Model	UCD	7
4	By-paragraph	Model	CD	7
5	Whole	List	UCD	3
6	Whole	List	CD	2
7	By-paragraph	List	UCD	15
8	By-paragraph	List	CD	8

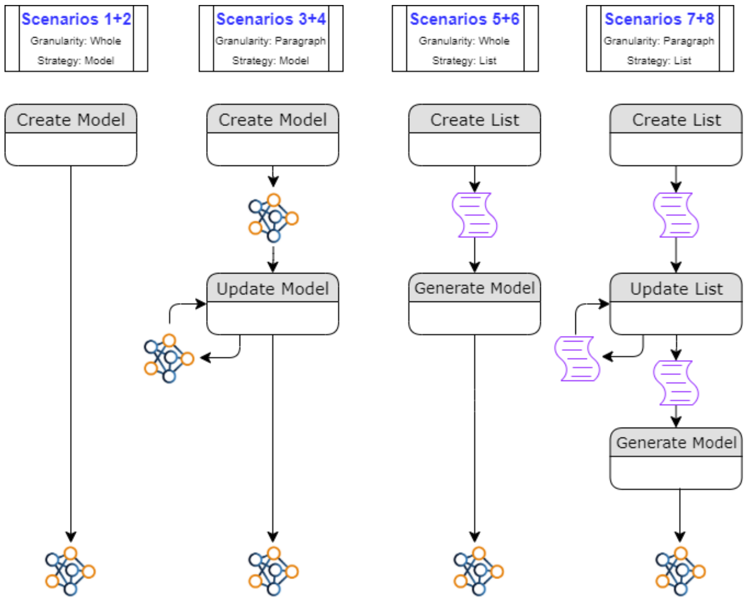

Fig. 2: The required LLM interactions for the different experimental scenarios

be presented in the common format of PlantUML, while we did not provide a concrete format for the lists. Table 3 provides an overview of the templates, including their descriptions and format.

Fig. 3 illustrates the model generated by GPT for Scenario 4 in the Perfect Trip (PTr) domain. The LLM (GPT in this case) was asked to create and refine (update) a class diagram based on the sequential input of each paragraph from the PTr description. The resulting model, which considers all seven paragraphs, effectively identifies core relevant classes such as *User*, *Member*, *Trip*, *Place*, *Visit*, *Opinion* (interpreting Review), and *Recommendation*. However, it also includes several irrelevant classes derived from the description. These classes,

Table 3: Templates Used for IS Modeling Tasks

Template	**Description**	**Prompt**
Create List	Create a list of elements (e.g., use cases, actors, classes) based on the description.	Create a list of <*elements*> from the following description. The description: <*desc*>
Update List	Modify or refine an existing list of elements to incorporate new information or correct errors expressed in the concern.	Modify the list of <*elements*> to address the following concern. The concern: <*conc*>
Generate Model	Create a model (e.g., use case diagram or class diagram) from the list of elements generated for previous prompts.	Generate a <*model*> from the list of <*elements*> generated in the previous prompt. Present the response in the following format: PlantUML.
Create Model	Create a model directly from a given description.	Create a <*model*> from the following description. Present the response in the following format: PlantUML. The description: <*desc*>
Update Model	Modify an existing model to address a specific concern.	Modify the <*model*> to address the following concern. Present the response in the following format: PlantUML. The concern: <conc>

while mentioned in the text, pertain more to the system's functionality and are better suited for inclusion in a use case diagram rather than a class diagram. Examples of such misclassified elements include *MZ System*, *VP of Content*, *VP of Culture*, and *World Tourism Organization*, which are actors interacting with the system, as well as *Search*, which represents a potential use case. The model has further inaccuracies regarding attributes, associations, and missing classes.

3.3 Evaluation Procedure

The final output for each scenario, representing the last modeling step, was evaluated by two undergraduate students who had previously served as teaching assistants (TA) in an IS modeling course. Each TA assessed the outputs of one to two application domains. The evaluation criteria included both semantic and syntactic criteria:

- **Requirements Satisfaction**: The extent to which the outcomes align with the requirements outlined in the provided descriptions, including both *correctness* and *completeness.*
- **Redundancy**: The identification of unnecessary elements or relationships that were not mentioned in the descriptions. This can be partially associated with LLMs' *hallucination.*

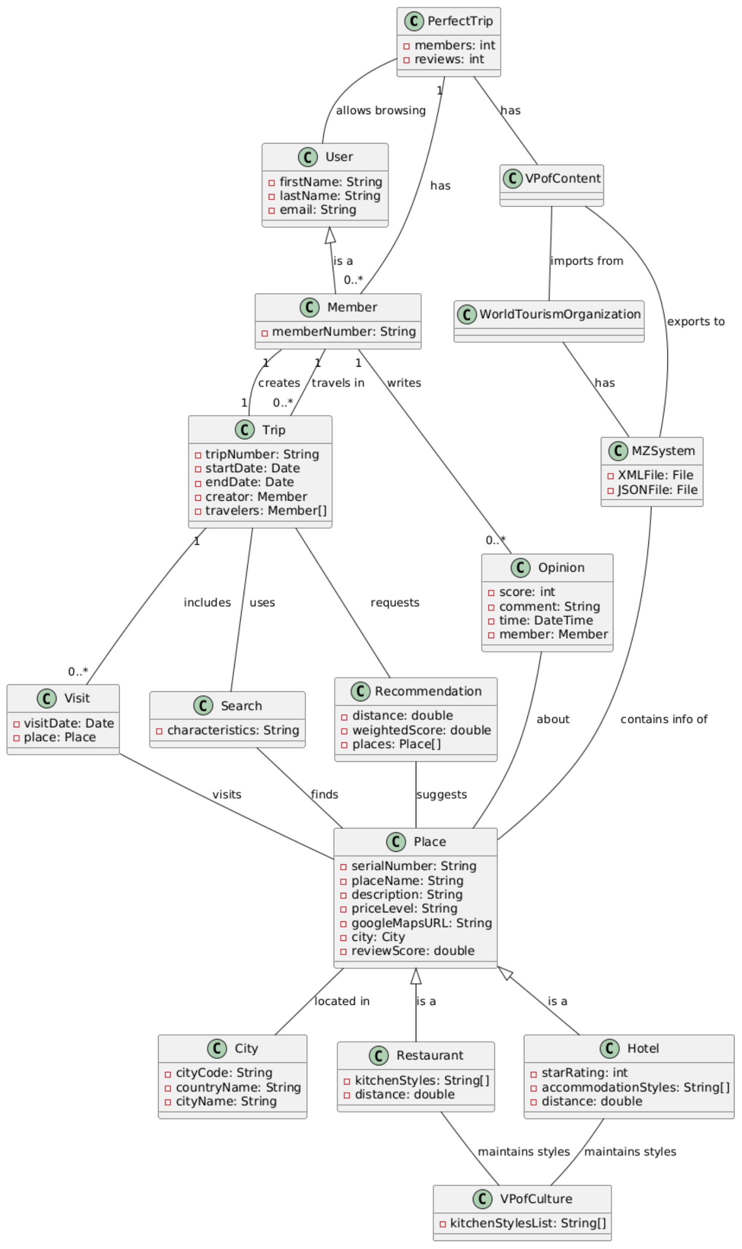

Fig. 3: A model generated by GPT for Scenario 4 in the PTr domain

– **Syntactic Correctness**: Identification of errors in the syntax of the model (UCD or CD) or in the PlantUML format.

To assess satisfaction of requirements, we first extracted model fragments from the descriptions, rather than generating complete reference solutions. This approach provides flexibility and allows for variations in the solutions. The extracted model fragments were of three types: use case fragment (including its actors), class fragment (including its attributes and operations), and association fragment (including the associated classes and potential attributes of the association class). Table 4 exemplifies part of the PTr description that is relevant to the creation of a class diagram, along with the expected model fragments.

Table 4: Examples of Specification and Expected Fragments for the Perfect Trip Domain

Specification	**Expected Fragments**

Perfect Trip's repository includes information on places of interest in different cities of the world. For each place, the file contains a unique serial number, the name of the place, a description, the price level (high/medium/low), the corresponding landmark on Google Maps expressed as a URL, and the city in which the place is located.	**class**.Place;
	attribute.serialNumber;
	attribute.placeName;
	attribute.description;
	attribute.level;
	attribute.linkToGoogleMaps;
	operation.calcWeightedScores
	class.PriceLevel;
	value.high; value.medium; value.low
	association.City-Place

City codes are unique and composed of the country number and a distinctive combination of three letters representing the city's name. For example, the city code for Tel Aviv is 972TLV, where 972 denotes the State of Israel's code.	**class**.City;
	attribute.cityCode;
	attribute.cityName
	class.Country;
	attribute.countryCode;
	attribute.countryName
	aggregation.Country-City

Restaurants and hotels are special kinds of places.	**class**.Hotel; inherits.Place
	class.Restaurant; inherits.Place

For each restaurant, the relevant kitchen styles should be kept.	**class**.Kitchen style;
	attribute.styleNumber;
	attribute.styleName
	association.Kitchen style-Restaurant

For each hotel, besides the previously mentioned details, we aim to include its star rating (ranging from 0 to 5), supported accommodation styles (AI - all-inclusive, BB - bed & breakfast, HB - half board, FB - full board and RO - room only). Note that a hotel may offer some or all accommodation styles.	**class**.Hotel;
	attribute.starRating;
	attribute.AI;
	attribute.BB;
	attribute.HB;
	attribute.FB;
	attribute.RO

With the extracted model fragments in hand, the evaluators were asked to assess the final models generated by the LLMs by scoring them against the various requirements and providing detailed comments on their assessments. To ensure the quality of the evaluations, two of the authors of this paper reviewed a sample of four models each (eight in total) and approved the TAs' assessments. The evaluation of the model in Fig. 3, based on the *partial list* of requirements outlined in Table 4, identified a missing enumeration class for the price level. Overall, the model achieved a requirements satisfaction score of 71.2%, a redundancy rating of 1 (very high), and a syntactic correctness score of 0 (no issues identified).

4 Results

The results were aggregated, such that each outcome (i.e., scenario per LLM) got a score on each evaluation criterion and analyzed to determine the impact of the experimental factors on model quality. As each criterion respond to multiple RQs, we indicate the related RQ in brackets within the subsequent text.

4.1 Requirements Satisfaction Results

Our scores for requirements satisfaction results for all experimental factors range from 15.4% to 98.6% as shown in Fig. 4. This wide range highlights the variability in LLM performance under different conditions. The Kolmogorov-Smirnov tests confirmed that the data adhered to normality assumptions. Consequently, t-tests were performed, showing no significant differences in mean scores for granularity (by-paragraph vs. whole, p = 0.6409) [**RQ1**], modeling strategy (list vs. model, p = 0.3472) [**RQ2**], or LLM (GPT vs. Llama, p = 0.7035) [**RQ3.1**]. ANOVA tests on domain (WSc, PTr, R4A) showed no significant effect on the requirements satisfaction score (p = 0.8794) [**RQ3.2**]. However, a significant difference was found between tasks (CD vs. UCD, p = 0.0013), with UCD showing higher mean scores [**RQ3.3**].

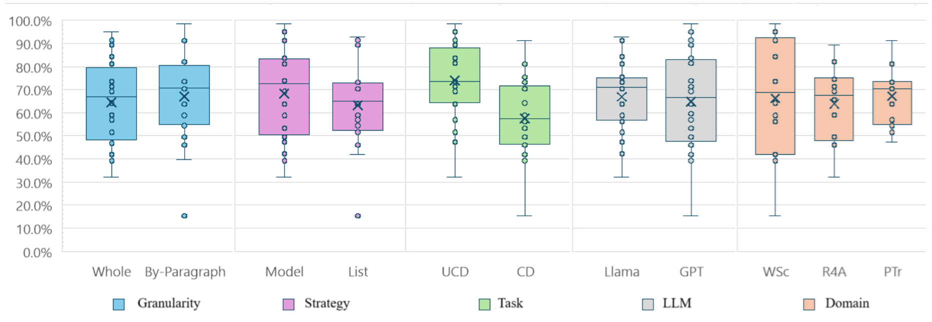

Fig. 4: Results of Requirements Satisfaction

A deeper analysis of requirements with low average scores (below 70%) across all scenarios or high standard deviations (above 33%) highlighted several constructs that posed challenges for the LLMs. In use case modeling, the most prominent issues were identifying *include* and *extend* dependencies and misinterpreting system boundaries. The latter often led to missing actors, use cases, or associations between specific actors and use cases [**RQ3.3**]. In class modeling, the LLMs struggled particularly with association classes, inheritance relationships, and enumeration types. These findings are aligned with other studies in the literature, as reviewed in Section 2.2. Common errors included overlooking or misplacing attributes of associations, incorrectly identifying super- and subclasses, and mislocating attributes of super-classes. Additionally, attributes of enumeration types were frequently misidentified as string attributes.

4.2 Redundancy and Syntactic Correctness Results

Wilcoxon tests revealed significant differences in Redundancy between granularity levels (Z = -2.4392, p = 0.0186) [**RQ1**] and tasks (Z = -2.3629, p = 0.0223) [**RQ3.3**]. Granularity at the by-paragraph level resulted in higher redundancy compared to the whole-description approach. This observation can be explained by the partial context provided to the LLM in the by-paragraph granularity, which “encourages” it to supplement or invent information that may already exist or be introduced differently in subsequent paragraphs. The segmented processing may lead the LLM to produce redundant content to ensure comprehensiveness.

For tasks, class modeling (CD) exhibited greater redundancy than use case modeling (UCD). This can be attributed to the inherently abstract and structural nature of CD tasks, which often involve defining and describing relationships, hierarchies, and attributes in detail. To ensure the correctness of these complex constructs, the LLM may over-communicate or reiterate information. In contrast, UCD tasks tend to have more explicit requirements, providing clearer guidance and reducing opportunities for redundancy [**RQ3.3**].

Syntactic errors were observed across all combinations of Granularity, Strategy, LLM, and Domain. Many of these errors stemmed from the LLM’s difficulty in distinguishing between information relevant to use case modeling (UCD) and class modeling (CD). Consequently, some outputs improperly combined classes with use cases or actors, resulting in a violation of the PlantUML syntax. The most extreme instance occurred in Scenario 3, where Llama produced a model containing only classes and actors, despite the task being to create a use case diagram. Additional syntactic errors included outputs where actors lacked corresponding use cases and vice versa, as well as comments in parentheses being misinterpreted as operations rather than attributes. These issues highlight the LLM’s struggle to maintain strict adherence to modeling conventions and emphasize the need for post-processing mechanisms to improve the generated models.

4.3 Discussion and Implications

In the following, we report on the implications of our findings to LLM-based domain modeling.

The severe limitations observed in LLM-generated outcomes regarding requirements satisfaction, redundancy, and syntactic correctness emphasize the need for active human involvement in the domain modeling process. Specifically, LLMs struggle with accurately identifying relationships, such as *include* and *extend* dependencies in use case diagrams, and *associations*, *inheritance* relations, and *association classes* in class diagrams. These challenges align with findings from other recent studies [8,5,19], highlighting that, despite their potential, LLMs are not yet capable of autonomously generating high-quality domain models. Consequently, practitioners must act as both facilitators and validators to align models with the intended requirements. This reliance on human involvement not only underscores the limitations of current LLMs but also calls for more targeted research for developing LLM-assisted (rather than LLM-generated) domain modeling approaches.

Implication I: LLM-assisted, rather than LLM-generated, domain modeling methods should be developed to enable iterative interactions with the LLM and incorporate validation processes to enhance model accuracy and ensure alignment with requirements.

The findings also suggest that while by-paragraph granularity does not significantly affect requirements satisfaction, it does increase redundancy, raising concerns about LLMs' ability to handle large, complex domain descriptions. This highlights the need for improved methods that support engineering prompts for real-world domain descriptions, which are typically multifaceted and more extensive. Our findings further show that if the descriptions include information relevant to different aspects of the domain, such as the domain functionality and structure, the LLMs often become "confused" and include incorrect elements (e.g., placing classes in use case diagrams or actors in class diagrams). Additionally, they sometimes attempt to enforce the creation of model elements, like representing actors or use cases as classes, as shown in the example in Fig. 3.

Implication II: Domain descriptions can be split and provided sequentially to LLMs, which may have a marginal effect on requirements satisfaction but could significantly impact redundancy. This requires careful consideration of how the descriptions are split to minimize redundancy and maintain model correctness.

The lack of significant differences in the mean scores for the strategy variable (list-based vs. model-based) suggests that the choice of modeling strategy does not substantially impact the correctness of LLM-generated models. This finding implies that both strategies are similarly effective in guiding LLMs to produce correct domain models. However, this result also highlights the flexibility of LLMs in adapting to different approaches without significant performance variation, which could be advantageous for modelers with varying preferences or expertise. Nonetheless, future research could explore other strategies and different qualities, such as usability, scalability, or time efficiency.

Implication III: Model-based and list-based modeling strategies result in models of similar quality, allowing modelers to choose the approach that best suits their preferences and expertise.

4.4 Threats to Validity

The discussion of threats to the validity of this research is aligned with the major threat categories introduced by Wohlin et al. [23]. With respect to *conclusion validity*, our results are limited to the scope and extent covered by our scenarios. In total, we focused on eight scenarios (cf. Table 2) which we applied to three distinct application domains using two LLMs. In total, this resulted in 48 models being created through the execution of 132 prompts.

Construct validity threatens the validity of our assessment of the LLM-generated models, as the evaluation was primarily conducted by graduate students. To address this, we briefed the students, who were already experienced in evaluating modeling tasks through their role as teaching assistants. We further prepared them by providing expected model segments, offering feedback on their initial evaluations, and guiding them on any questions they had. Additionally, two authors of this paper reviewed and approved samples of their evaluations to ensure the quality of the assessment.

We do not see a strong human-focused threat regarding the *internal validity* as we prompted the LLMs systematically with identical prompts. Unlike many other studies, we did not rely on human modelers or prompters.

Finally, to address *external validity* concerning the generalizability of our findings, we used three different application domains to ensure that the results are not incidental. This was intended to enhance the potential transferability of our conclusions. The chosen domains were common and expected to be familiar to LLMs. Furthermore, the fact that we used two LLMs, amongst the vast array of LLMs from different vendors, can be an external validity threat in our work. However, we chose two state-of-the-art LLMs to mitigate this threat. Moreover, while we acknowledge that the rapid development of LLMs poses a threat that our results will soon become outdated, it is important to emphasize that advances in LLMs do not necessarily translate into corresponding improvements in domain modeling. The primary focus of LLM development lies in enhancing general reasoning capabilities, rather than specifically targeting domain modeling or model-driven engineering. Although improved reasoning can support domain modeling tasks, existing studies have shown that even the substantial leap from GPT-3.5 to GPT-4 yielded comparable results and revealed similar limitations in this context [19,8]. This suggests that achieving qualitatively different results may require training LLMs specifically for domain modeling tasks.

5 Conclusion

The use of LLMs for domain modeling is still in its early stages, with initial, explorative results. A systematic and maturated approach to utilizing LLMs in modeling is still greatly needed. In this paper, we presented our investigation into the impact of different levels of granularity (whole text vs. by-paragraph), modeling strategies (model-based vs. list-based), and tasks (use case modeling vs. class modeling) on the quality of LLM-generated domain models, with a particular focus on semantic and syntactic correctness. Our results demonstrated that state-of-the-art LLMs produce domain models of varying quality, with significant findings regarding the impact of granularity on redundancy (by-paragraph resulted in more redundancies) and the effect of task on requirements satisfaction (class diagrams resulted in less satisfied requirements). The value of this paper goes beyond these experimental findings, by providing a comprehensive discussion and raising implications for LLM-assisted domain modeling. These research challenges must be addressed before LLMs can be considered co-modelers in the domain modeling process – similar to the role Copilot plays in software engineering.

Future research includes improving LLM-assisted domain modeling methods to handle larger and more complex domain descriptions. This requires developing more effective strategies for interactions and integrating robust validation processes to enhance model quality. Furthermore, exploring the integration of LLMs with human modelers in a collaborative co-creation environment will be crucial for advancing LLM-assisted modeling. Research into new prompt engineering techniques that minimize redundancy and improve model correctness will also be key to ensuring that LLMs can be effectively applied in real-world domain modeling scenarios. In addition, we plan to explore the assistance of LLMs in behavioral modeling tasks, such as generating state diagrams, to assess their potential beyond functional and structural modeling. Finally, we plan to explore the process of LLM-assisted domain modeling by analyzing both intermediate and final generated models, as well as the paths explored during the modeling process.

Acknowledgment: The authors would like to sincerely thank Neta Mandelbaum and Matan Meirovitz for their invaluable assistance in evaluating the LLM-generated domain models.

References

1. Ali, S.J., Reinhartz-Berger, I., Bork, D.: How are llms used for conceptual modeling? an exploratory study on interaction behavior and user perception. In: Maass,

W., Han, H., Yasar, H., Multari, N.J. (eds.) Conceptual Modeling - 43rd International Conference, ER 2024, Pittsburgh, PA, USA, October 28-31, 2024, Proceedings. Lecture Notes in Computer Science, vol. 15238, pp. 257–275. Springer (2024). https://doi.org/10.1007/978-3-031-75872-0_14

2. 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)
3. Bajaj, D., Goel, A., Gupta, S., Batra, H.: Muce: a multilingual use case model extractor using gpt-3. International Journal of Information Technology **14**(3), 1543–1554 (2022)
4. Burgueño, L., Clarisó, R., Gérard, S., Li, S., Cabot, J.: An nlp-based architecture for the autocompletion of partial domain models. In: Advanced Information Systems Engineering - 33rd International Conference, CAiSE 2021, Australia, 2021. vol. 12751, pp. 91–106. Springer (2021). https://doi.org/10.1007/978-3-030-79382-1_6
5. Cámara, J., Troya, J., Burgueño, L., Vallecillo, A.: On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml. Software and Systems Modeling **22**(3), 781–793 (2023)
6. Chen, B., Chen, K., Hassani, S., Yang, Y., Amyot, D., Lessard, L., Mussbacher, G., Sabetzadeh, M., Varró, D.: On the use of gpt-4 for creating goal models: an exploratory study. In: 2023 IEEE 31st International Requirements Engineering Conference Workshops (REW). pp. 262–271. IEEE (2023)
7. Chen, B., Yi, F., Varró, D.: Prompting or fine-tuning? a comparative study of large language models for taxonomy construction. In: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). pp. 588–596. IEEE (2023)
8. Chen, K., Yang, Y., Chen, B., López, J.A.H., 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). pp. 162–172. IEEE (2023)
9. 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) **18**, 1–15 (2023)
10. Francu, J., Hnetynka, P.: Automated generation of implementation from textual system requirements. In: Software Engineering Techniques - Third IFIP TC 2 Central and East European Conference, CEE-SET, Czech Republic, 2008. Lecture Notes in Computer Science, vol. 4980, pp. 34–47. Springer (2008). https://doi.org/10.1007/978-3-642-22386-0_3
11. Herchi, H., Abdessalem, W.B.: From user requirements to UML class diagram. CoRR **abs/1211.0713** (2012), http://arxiv.org/abs/1211.0713
12. Jahan, M., Hassan, M.M., Golpayegani, R., Ranjbaran, G., Roy, C., Roy, B., Schneider, K.: Automated derivation of uml sequence diagrams from user stories: Unleashing the power of generative ai vs. a rule-based approach. In: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems. pp. 138–148 (2024)
13. Kim, J., Park, S., Jeong, K., Lee, S., Han, S.H., Lee, J., Kang, P.: Which is better? exploring prompting strategy for llm-based metrics. arXiv preprint arXiv:2311.03754 (2023)

14. Moody, D.L., Sindre, G., Brasethvik, T., Sølvberg, A.: Evaluating the quality of information models: Empirical testing of a conceptual model quality framework. In: Clarke, L.A., Dillon, L., Tichy, W.F. (eds.) Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA. pp. 295–307. IEEE Computer Society (2003). https://doi.org/10.1109/ICSE.2003.1201209
15. Nelson, H.J., Poels, G., Genero, M., Piattini, M.: A conceptual modeling quality framework. Softw. Qual. J. **20**(1), 201–228 (2012). https://doi.org/10.1007/S11219-011-9136-9
16. Reinhartz-Berger, I., Ali, S.J., Bork, D.: Leveraging llms for domain modeling: The impact of granularity and strategy on quality - online supplementary material (Apr 2025). https://doi.org/10.5281/zenodo.15192220
17. Robeer, M., Lucassen, G., van der Werf, J.M.E.M., Dalpiaz, F., Brinkkemper, S.: Automated extraction of conceptual models from user stories via NLP. In: 24th IEEE International Requirements Engineering Conference, RE China, 2016. pp. 196–205. IEEE Computer Society (2016). https://doi.org/10.1109/RE.2016.40
18. Saini, R., Mussbacher, G., Guo, J.L.C., Kienzle, J.: Machine learning-based incremental learning in interactive domain modelling. In: 25th International Conference on Model Driven Engineering Languages and Systems, MODELS, Canada, 2022. pp. 176–186. ACM (2022). https://doi.org/10.1145/3550355.3552421
19. Silva, J., Ma, Q., Cabot, J., Kelsen, P., Proper, H.A.: Application of the tree-of-thoughts framework to llm-enabled domain modeling. In: Maass, W., Han, H., Yasar, H., Multari, N.J. (eds.) Conceptual Modeling - 43rd International Conference, ER 2024, Pittsburgh, PA, USA, October 28-31, 2024, Proceedings. Lecture Notes in Computer Science, vol. 15238, pp. 94–111. Springer (2024). https://doi.org/10.1007/978-3-031-75872-0_6
20. Weyssow, M., Sahraoui, H.A., Syriani, E.: Recommending metamodel concepts during modeling activities with pre-trained language models. Softw. Syst. Model. **21**(3), 1071–1089 (2022). https://doi.org/10.1007/S10270-022-00975-5
21. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382 (2023)
22. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint arXiv:2303.07839 (2023)
23. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering, Second Edition. Springer (2024). https://doi.org/10.1007/978-3-662-69306-3
24. Yang, S., Sahraoui, H.A.: Towards automatically extracting UML class diagrams from natural language specifications. In: 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS, Canada, October 2022. pp. 396–403. ACM (2022). https://doi.org/10.1145/3550356.3561592