Towards Evaluating Trustworthy Modeling Assistants

Vilmos Bilicki[1], Márton Elekes[1,*], Kristóf Marussy[1] and András Vörös[1]

[1]*Budapest University of Technology and Economics, Faculty of Electrical Engineering and Informatics, Department of Artificial Intelligence and Systems Engineering, Budapest, Hungary*

Abstract

Comprehending and navigating large engineering or business models requires specialized expertise, which is costly and limits the prevalence of modeling. LLM-based agents are increasingly used to perform various tasks in large codebases. In order for such agents to be trustworthy, it is crucial that their answers are grounded in existing model elements. While different agent architectures have been proposed for assisting modeling tasks, their rigorous and automatic evaluation has to rely on domain-specific performance metrics and large corpora of benchmark models and example user questions. Therefore, their application in modeling environments remains limited by the lack of publicly available model and benchmark sets.

This paper presents an extensible benchmark to support evaluation during the development of LLM-based agents for modeling domains based on publicly available engineering and business models from two domains and corresponding questions provided by domain experts. We propose a workflow for question augmentation and compiling reference answers and relevant modeling elements with LLM. Using the framework makes LLM modeling agents comparable by several state-of-the-art metrics from the literature and gives fast feedback during their development. We demonstrate the usage of the benchmark framework by evaluating our LLM agents which use ontological knowledgebase and SPARQL queries to assist modeling in safety- and business-critical domains.

Keywords

benchmark, Model-Based Systems Engineering, modeling assistant

1. Introduction

Comprehending large engineering models and navigating in them require specialized expertise, which is costly and limits its use to only specific, critical domains. LLM-based agents are increasingly used to perform various tasks in large codebases. However, their application in Model-Based Systems Engineering (MBSE) remains limited due to the following challenges:

1. Committing errors in modeling can have significant cost, therefore, trustworthy and grounded agents are required for this field.
2. The engineering models often have large size and high complexity, and are stored in specialized *model stores* instead of the file system.
3. Modeling languages are *low-resource* programming languages since publicly available models are scarce as they are used in specialized development processes mostly for critical systems.

In this work, we investigate **trustworthy modeling assistants** (Fig. 1) that answer textual questions from *agent users* (★☆☆). To overcome Challenge 1 and increase the *reliability* of *textual answers* produced by the assistant, the agent provides *grounding from model elements* as a means of *verifying* the correctness of answers. The user may navigate to the referenced model elements in the model store or editor to inspect them along with the answer. To produce answers and determine the appropriate grounding

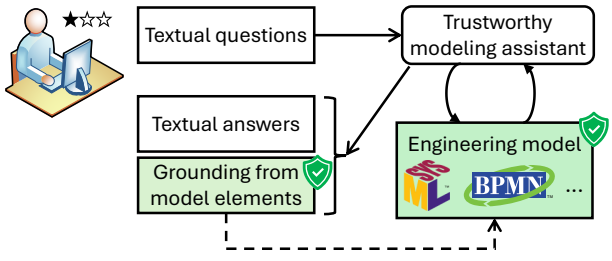


Figure 1: Trustworthy modeling assistant

Joint Proceedings of the STAF 2026 Workshops: AgileMDE, GCM, ICMM, LLM4SE, TTC. Rennes, France, June 29-July 3, 2026

*Corresponding author.

bilicki.vilmos@edu.bme.hu (V. Bilicki); elekes@mit.bme.hu (M. Elekes); marussy@mit.bme.hu (K. Marussy); vori@mit.bme.hu (A. Vörös)

0000-0002-3558-147X (M. Elekes); 0000-0002-9135-8256 (K. Marussy); 0000-0001-7617-3563 (A. Vörös)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-26

CEUR-WS.org/Vol-4338/LLM4SE-paper5.pdf

model elements, the assistant has to interact with the *engineering model*. Therefore, implementations of trustworthy modeling agents have to address Challenge 2 to access the model store and Challenge 3 to effectively navigate it. This paper focuses on formal, graph-based, structural and behavioral models.

Due to Challenge 2, *modeling experts* (★★☆) developing a trustworthy modeling assistant have to tailor it for the particular modeling environment and model store. Meanwhile, Challenge 3 demands considerable *domain expert* (★★★) time and effort to curate a large corpus of high-quality *example questions* and an *evaluation oracle* (e.g., *ground truth answers* along with grounding model elements) to evaluate and fine-tune the assistant.

Semantic technologies are increasingly used in modern system engineering languages for integrating models and tools for various viewpoints, and they are also a promising area for providing grounded knowledge for LLMs. We aim to leverage these technologies to construct trustworthy modeling assistants and evaluate them. In particular:

- To tackle Challenge 2, we propose an *AI agent template* using an ontological knowledge base and SPARQL queries to assist modeling in various modeling domains. The template can be instantiated with various Large Language Models (LLM) for a capability/cost tradeoff.
- We propose a workflow for constructing corpora of benchmark questions and grounded answers based on *seed questions* from *domain experts* (★★★), and LLM-generated *question set augmentation* and *question mutation*. *Ground truth* answer generation involves the AI agent template instantiated with a high-capacity reasoning LLM and *modeling expert* (★★☆) in the loop to reduce reliance on *domain expert* (★★★) effort in Challenge 3.
- We construct a benchmark set based on *large-scale system engineering model*, the Thirty Meter Telescope (TMT) model [1, 2], as well as a case study from *business process engineering*. We explicitly reuse existing semantic mappings [3] showcasing the semantic integration.
- We propose an evaluation harness that implements state-of-the-art text-based metrics for answers and model element comparison for evaluating the quality of answer grounding.
- We evaluate and compare instantiations of the agent template with low-cost reasoning LLMs with the benchmark set and harness.

Our work enables using existing semantic mappings for LLM agents to help navigation in large system engineering models for critical systems, as well as business processes. The benchmark set, evaluation harness, and measurement results are publicly available on Github[1].

2. Related work

Using LLMs for engineering models has been investigated since they first appeared. Fill et al. [4] studied LLMs in domain-specific and software modeling, which promises more maintainable applications than directly generating code from textual specification. These approaches are hindered by syntax and semantic errors in LLM-generated software models [5], especially in complex models combining several modeling elements [6]. The main reason lies in the fact that modeling languages are low-resource programming languages [7] since publicly available models are scarce; therefore, LLMs could not incorporate the syntax and semantics during training. Therefore, LLMs can easily generate code snippets in popular programming languages, while using LLMs for niche languages or for models of complex software/system architectures requires different strategies. Model generation and following best practices and patterns to build maintainable and extendable models poses a challenge to LLMs [8].

Yu et al. [9] study industrial use-cases of generative AI applications and the quality measurements applied. They found that high-precision metrics are essential for practitioners to understand and quantify GenAI system limitations, as well as track the evolving performance, and decide on the acceptable quality. They identified that GenAI systems and evaluation tools lack the sufficient level of context awareness to align with task-/company-specific domain, which requires humans to review the outputs, limiting the adoption of GenAI. We suggest an efficient benchmark corpus generation workflow and an evaluation harness with automatic metrics to enable continuous, domain-specific

[1] https://github.com/ftsrg/mbse-assistant/tree/llm4se-2026

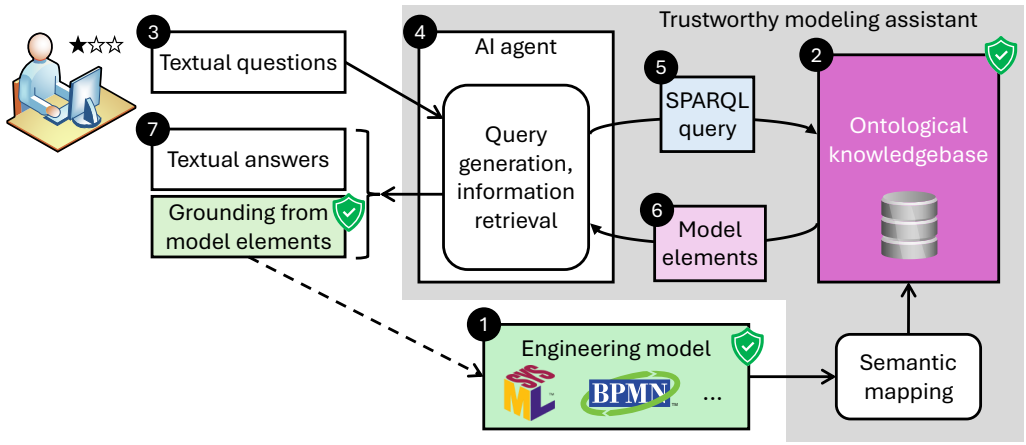


Figure 2: Trustworthy modeling assistant architecture template

evaluation of the performance. Their work highlights the importance of a multi-faceted approach to quality measurement, e.g., a combination of human, automated, and AI-assisted evaluation; therefore, we use both AI-assisted and human effort for the benchmark corpus, as well as automated and AI-based metrics for the continuous evaluation.

Knowledge bases are well suited to store models as graphs. SPARQL queries can be used for retrieving data from knowledge graphs. SPARQL queries can also be generated by LLM models since the language is sufficiently covered in LLM training corpora [10]. Karagoz et al. [11] use knowledge graph representations for analyzing system models with graph convolutional networks.

Civili et al. [10] discuss that not only the data in a knowledge graph but also its schema, the ontology can be too large to fit into the LLM context window. They argue that using a large, enterprise ontology to help LLM-based database query generation can be achieved by iteratively revealing only those parts of the ontology which are required to formulate the SPARQL query to answer the natural language question. Our approach uses few-shot prompting to show some modeling concepts and query examples in addition to providing a list of all classes and possible properties in the model.

Dunn [12] investigates LLM performance on 140 tasks including discovery, attribute extraction, 2-5 hop navigation and description generation on SysML v2 models. Their benchmark set contains 6 SysML v2 models (80-400 KB each) and a knowledge graph toolkit for SysML v2 with command-line and Model Context Protocol (MCP) interface[2]. They found that the performance drastically decreases on the largest model (400 KB), even with graph navigation and vector search tools. Our benchmark contains larger models, the SysML model is 39 MB and the BPMN model is 3 MB in size.

3. Ontological modeling agent template

We propose an *architecture template* for a trustworthy modeling assistant (Fig. 2), which can be considered an implementation of the general trustworthy modeling assistant concept from Fig. 1.

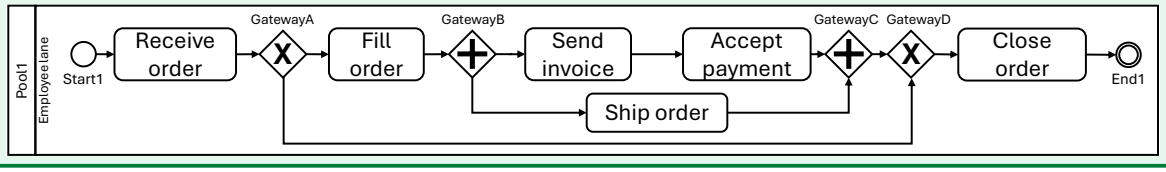
Semantic mapping. The ❶ *engineering model* is first imported into an ❷ *ontological knowledge base* in the *semantic mapping* step to enable uniform access by the assistant. The RDF representation is built on *semantic triples*, there both concepts from the model (i.e., model elements) and the metamodel (i.e., classes and references) are represented as URIs to present a unified interface for LLMs. In our implementation, we used Apache Fuseki as the triple store.

The import step explicitly reuses existing mappings to RDF for handling modeling languages, such as SysML and BPMN. Thus, Challenge 2 can be addressed by writing a suitable system prompt, and reusing or developing a semantic mapping from the source modeling language to RDF.

Agent loop. The *user* (★☆☆) interacts with the assistant by asking a ❸ *textual question*. The assistant

[2] https://gitlab.com/nomograph/sysml

(a) ❶ Engineering model being queried in BPMN format



(b) ❷ Knowledge base in RDF Turtle format

```
@prefix bpmn:
  <https://example.com/bpmn/> .
@prefix model:
  <https://example.com/model/> .
model:GpiHsM69EfCgr
  rdfs:comment "A Sequence Flow is..." ;
  rdfs:subClassOf bbo:sequenceFlow ;
  bpmn:has_sourceRef model:4fxXs868EfCgr ;
  bpmn:has_targetRef model:D9G9sM69EfCgr .
# ...
```

(c) ❸ Textual question

How many tasks are included in the model?

(d) ❹ Agent system prompt

You are an expert assistant that converts natural language questions about BPMN choreography models into SPARQL queries. The models are stored as RDF triples in a knowledge graph...

(e) ❺ Generated SPARQL query

```
SELECT ?elem ?name ?subClass ?comment
WHERE { ?elem bpmn:name ?name .
  ?elem rdfs:subClassOf ?subClass .
  OPTIONAL { ?elem rdfs:comment ?comment }
  FILTER(CONTAINS(LCASE(STR(?subClass)),
    "task")) } LIMIT 20
```

(f) ❻ SPARQL query results

elem	name	subClass	comment
model:D9G9sM69EfCgr	Receive order	Task	A Task is a …
model:iN02NW3421Evb	Fill order	Task	A Task is a …
…	…	…	…

(g) ❼ Textual answer

There are 6 tasks in the model.

(h) Grounding model elements

Receive order, Fill order, Send invoice, Accept payment, Ship order, Close order

Figure 3: Business process case study in BPMN

is realized as an ❹ *LLM agent loop* with *tool use*. The ❺ *SPARQL query* tool executes queries on the data store. We selected SPARQL as the query language, as LLMs have been shown to be highly capable of utilizing it to interact with ontological knowledge bases, especially if they are provided with a description of the underlying ontology [10]. The tools returns the resulting ❻ *model elements* and other query results as a JSON array to the agent. To reduce context bloat, we limit all queries to up to 20 result rows. The *final answer* tool provides the ❼ *textual answer* to the user.

The LLM is invoked in a loop, executing the generated SPARQL queries, until the final answer tool is called, which terminates the loop. If the generated query is syntactically invalid (2% in this setup), the LLM can use the error message to repeat query generation. The agent is also extensible by incorporating other tools, e.g., documentation search via Retrieval-Augmented Generation (RAG). However, in the rest of this paper, we assume that the assistant has been instantiated with the two aforementioned tools.

Trustworthy answers. The assistant extracts *grounding from model elements* from the result set of the last *SPARQL query*, which we assume to be the basis of the final answer. Questions requiring more complex reasoning and not answerable from model elements are out of scope. If the final SPARQL query contained variables in its SELECT clause that matched model element URIs, the returned URIs are used for supporting the answer. Otherwise, we use the URIs matched by any variable in the query body.

Mapping the support URIs to model element identifiers relies on the traceability information from the semantic mapping. For example, the semantic mapping may encode element UUIDs or other identifiers in the URIs. Therefore, the user can open the model elements directly inside the modeling environment or editor to inspect them and decide whether they support the textual answer. Thus, divergences caused by possible errors in the semantic mapping can also be discovered.

Example 1. *Fig. 3 shows an example execution of the agent in the business process modeling domain.*

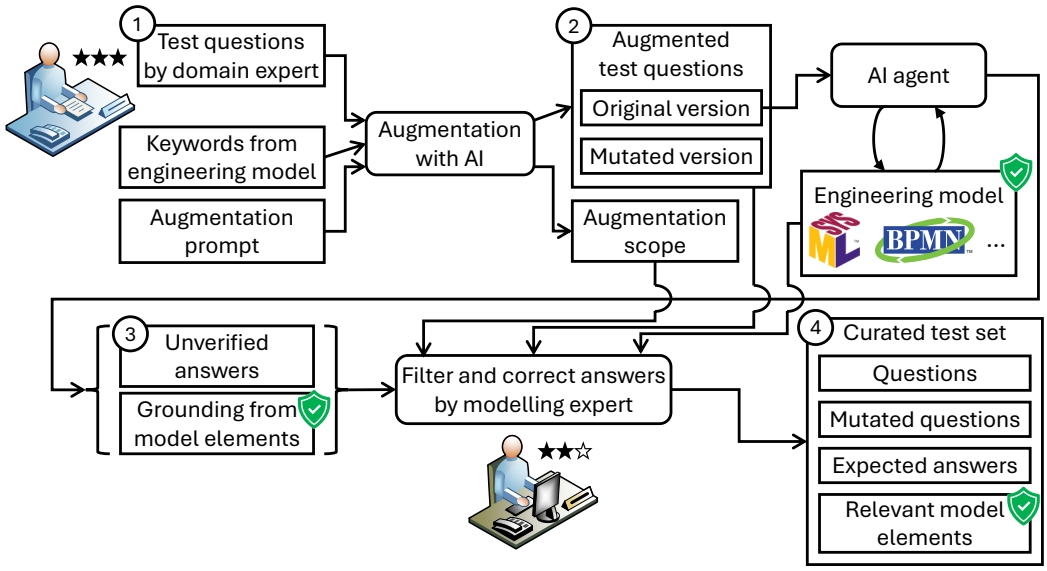


Figure 4: Benchmark set augmentation workflow

We used the semantic mapping from [3] to import the BPMN model in Fig. 3a into the ontology shown in Fig. 3b in RDF Turtle format. Notice that the semantic mapping has added rdfs:comment annotations to each model element describing the semantic meanings of their classes. Fig. 3d shown excerpt from the system prompt *of the agent. The full system prompt also includes a description of the ontology.*

Fig. 3c shows a question by the user. In response, the agent has generated the SPARQL query shown in Fig. 3e. Note we provided few-shot *prompting query example to instruct the agent to filter class names using string comparison to allow matches even when it generate class names slightly different from those in the ontology, and it also queries the* `rdfs:comment` *to incorporate class descriptions into the resulting context. Fig. 3f shows an excerpt of the query results obtained from Apache Fuseki.*

After receiving the results, the agent has used the final answer tool to output the answer in Fig. 3g and terminate the agent loop. We extract the model element URIs from Fig. 3f to obtain the references to supporting model elements in Fig. 3h that can be opened by the user in a BPMN editor.

4. Benchmark set for trustworthy modeling agents

We now investigate the construction of benchmark sets for trustworthy modeling agents, such as those instantiated from Section 3.

A significant challenge in the *continuous benchmarking* of LLM-based agent is the automated evaluation on answer relevance and trust, as manual evaluation would demand considerable *domain expert* (★★★) or *modeling expert* (★★☆) time. Thus, while automated evaluation cannot fully replace manual evaluation [13], we aim to support continuous benchmarking by providing benchmark sets suitable for automated evaluation.

In particular, we import publicly available engineering and business models by semantic mapping, and construct a corpus of relevant questions and corresponding *ground truth* answers. The answers are also equipped with references to 🛡 *grounding model elements* that support them. Automated evaluation can use text-based metrics (Section 5.1) to estimate assistant accuracy by comparing the ground truth answers to the *AI agent under evaluation*. The quality of returned supporting model elements is also estimated by comparing them to the ground truth set.

4.1. Question set augmentation

Instead of relying on *domain expert* (★★★) effort to write a large set of sample questions and answers based on familiarity and experience with the models, we follow a *benchmark set augmentation* workflow

depicted in Fig. 4. The *domain expert* (★★★) writes a small set of ① *seed questions*. We run an LLM-based *question augmentation* procedure to generate further ② *augmented questions*. The *prompt* for the augmentation includes the seed questions and *keywords* extracted from the domain model (but not direct access to the model). Each seed question is annotated with a *scope* regarding its complexity and the parts of the model required for answering it.

Question scope. The *domain expert* (★★★) annotates seed question with a *scope* of knowledge required to answer it. The augmentation LLM generates *augmentation scope* annotations for the generated questions. We note that generated questions – similarly to expert written questions – may not necessarily be balanced across scopes, but the annotation may nevertheless serve as metadata when evaluating benchmark results. Scope annotations are withheld from the AI agent under evaluation.

Ground truth answers. We instantiate the assistant from Section 3 with an LLM with high reasoning capacity to generate ③ *unverified answers* based on *engineering model*. These answers and their links to supporting model elements provide a starting point for a *modeling expert* (★★☆) to review the engineering model and *filter* out invalid questions and *correct* the *expected answers* and *grounding model elements* according to the *augmentation scope*. In this step, we assume expertise with the modeling environment and modeling language semantics. However, the presence of the LLM starting point means that the modeling expert does not need to be deeply familiar with the whole engineering model; if correct grounding model elements cannot be found near the LLM suggested ones, they can simply filter out the question as possibly invalid. As a result, we obtain a ④ *curated benchmark set* of modeling expert in the loop reviewed questions, answers, and model elements.

A limitation of this approach is that while the *modeling expert* (★★☆) can review the answers for including incorrect model elements and hallucinations, modeling elements present in the engineering model but missing from the LLM suggestion cannot be easily determined. This is especially problematic when the augmentation scope is wide, i.e., we wish to generate questions that require understanding across the whole engineering model. Ground truth answers in such wide augmentations scopes should ideally be reviewed by a *domain expert* (★★★) at least in part.

Mutant question generation. We incorporate an additional benchmark set augmentation method from *metamorphic testing* [14]. We use an LLM with instructions to replace keywords and phrasing in generated questions without affecting the overall meaning. The *modeling expert* (★★☆) reviews the mutant questions to verify that the meaning was preserved. Ground truth answers and model elements remain applicable for the mutant, reducing the need for further *domain expert* (★★★) effort.

Example 2. *Consider the original question* "Based on the models of this organization, is there a sole profile of the organization?". *The augmentation scope is a specific organization owning multiple BPMN models. Metamorphic generation produces the mutant* "Can you determine a single organizational profile based on this organization's models?". *The expected answer is* "The organization is most likely a university course on Business Process Modeling using a travel agency booking scenario as an example.".

4.2. Evaluation domains

Business process modeling (BMPN). The publicly available *Signavio Academic Models* [15] is a large dataset of business process models from the *SAP Signavio* platform. We follow [16] and export 281 BPMN processes from the dataset. We use the BPMN to RDF transformation from [3] as a semantic mapping and construct a single joint ontology from all BPMN models with 13 646 distinct entities.

For augmentation scopes, we consider *general*, *organization-specific*, and *single-model* questions, depending on whether they inquire about all BPMN processes, processes connected to a single organization, or a single process, respectively. We consider questions focusing on either the *structure* of the process (e.g., the presence of cyclic behaviors, the number of tasks, etc.), or *domain-specific questions* that refer to terminology (e.g., task names, pool names, etc.) from the business domains.

We seeded the question set with 71 questions written by one of the authors. We generated 135 questions with Claude 4.6 Opus, of which 124 remained after modeling expert review (mostly removed duplications and irrelevant syntactic questions). We also generated 195 mutant questions. Single-process questions dominate both human-written (50) and LLM-augmented (100) questions, because

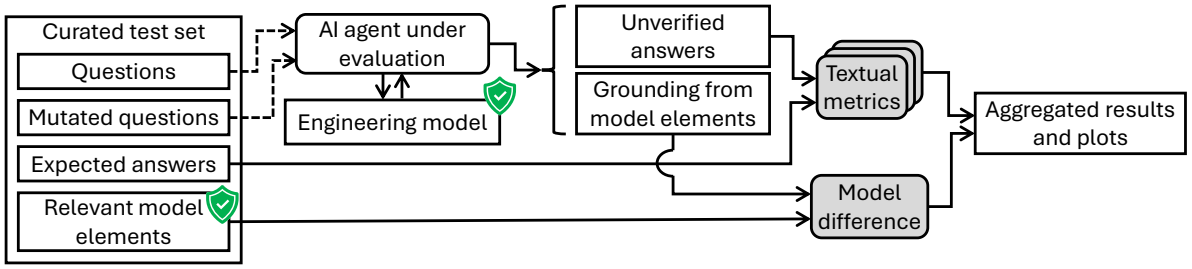


Figure 5: Modeling assistant evaluation harness

most organizations in the dataset own only a single process. We instantiated the assistant from Section 3 to generate answer candidates with Claude 4.6 Opus, which were reviewed and corrected by the author serving as the modeling expert for ground truth answers.

Systems engineering (SysML). The Thirty Meter Telescope (TMT) model [1, 2] is a publicly available, large SysML 1.6 model from NASA JPL. We reuse a SysML to RDF transformation[3] to construct an ontology from the model with 278 886 distinct entities.

An expert from JPL seeded the generation with 9 questions in three scopes: *junior-*, *medior-*, and *expert-level*. We generated 24 questions with GPT-5.2, of which 19 remained after review. We also generated 28 mutant questions. We instantiated the assistant from Section 3 with GPT-5.2 to generate answer candidates, which were reviewed and corrected by two authors of the paper serving as the modeling experts to obtain ground truth answers.

5. Evaluation

Fig. 5 shows our evaluation approach for trustworthy modeling assistants. The *AI assistant under evaluation* is executed on the *original* and *mutant* questions. We use a suite of *text-based metrics* to compare the answers to ground truth answers, while grounding model elements are compared to the ground truth by *precision* and *recall*. We consider various instantiations of the template from Section 3, but note that our harness can also evaluate other trustworthy modeling assistants independent of their internal architecture if they provide textual answers and the identifiers of the relevant model elements.

We aim to address the following research questions:

RQ1 How do various LLMs in the assistant instantiations compare in effectiveness?

RQ2 Which metrics are highly correlated?

5.1. Metrics for automatic comparison

The metric values with the exception of **J_1-5** are in the range [0,1], and higher values are better.

Keyword-based metrics. We assign to each ground truth answer a set of *keywords* and compute *keyword recall* for the textual answers of the assistant under evaluation, i.e., the fraction of ground truth keywords found in the answer. We consider two variations of this approach: *Exact matching* (**KW_EM**) counts only exact occurrences in the answer. *LLM-based matching* (**KW_LLM**) aims to mitigate false negatives caused by synonyms by using Opus 4.6 Claude. The system prompt instructs the model to also consider synonyms to find keyword occurrences.

N-gram based metrics. Two metrics commonly used in the Natural Language Processing (NLP) community for text similarity are **chrF** [17] and **ROUGE-L** [18]. In the case of chrF, we consider n-grams of length $n = 1, \ldots, 6$ as in [17], and for ROUGE-L, we set $\beta = 1$ as in [18].

Embedding-based similarity. *Text embedding* methods use neural networks to embed texts into vector spaces for estimating semantic similarity. The *cosine similarity* (**COS**) [19] metric computes normalized dot product $\frac{v_{gt} \cdot v_{out}}{|v_{gt}| \cdot |v_{out}|}$ of the embedding vectors of the ground truth answer v_{gt} and the

[3] https://github.com/koneksys/cameo2rdf

assistant answer v_{out}. We used `all-MiniLM-L6-v2` [20] as the embedding model, which produces 384-dimensional vectors. *BERTScore* (**BERT**) [21] is another popular embedding-based metric. It is based on the embeddings computed by the BERT language model (`bert-base-cased`) [22] for each token of the input texts. The score derives a similarity metric after the pairwise matching of the embedding vectors of tokens from the ground truth and output answers.

LLMs as judges. LLM-based evaluation of answers uses prompts to assign similarity scores to answers [23]. We implement two such scores using Claude 4.6 Opus as the judge, and include domain and question scope in the prompt to contextualize questions and answers to be judged.

The *2 out of 3* (**J_2o3**) generates 3 separate 'pass or fail' answers (i.e., whether the output answer matches the ground truth). The final score is derived by majority vote. We set the inference temperature to 0.2 to introduce some randomness to the votes. The *grading* (**J_1-5**) judge uses a single generation to assign a score between 1 to 5 to the answers, where 1 denotes unrelated answers, and 5 denotes a good match beween the ground truth and the output answers.

Model difference. To compare the set of supporting model elements in the ground truth answer M_{gt} and the assistant answer M_{out}, we compute both the *precision* (**PREC**) $\frac{|M_{gt} \cap M_{out}|}{|M_{out}|}$ and *recall* (**REC**) $\frac{|M_{gt} \cap M_{out}|}{|M_{gt}|}$. These metrics support text-based metrics by evaluating whether the assistant has found the relevant model elements by SPARQL querying before producing the final textual answer.

5.2. RQ1: Comparison of LLMs

Table 1 shows the LLMs selected for instantiating the assistant from Section 3 for evaluation along with their usage costs, the total number of tokens consumed during the evaluation, and the average number of SPARQL queries issued per BPMN and SysML question answering.

The rationale for selecting these LLMs was to construct assistants that are suitable for everyday use in a modeling environment with low costs. We note that these LLMs are much smaller and cheaper than the LLMs used during candidate answer generation for modeling expert review (GPT-5.2, Claude 4.6 Opus), where we selected the most capable LLMs available to reduce modeling expert review effort.

Table 1: LLMs evaluated in the benchmark

	$/1M		usage		avg. SPARQL	
	tok. in	tok. out	tok. in	tok. out	BPMN	SysML
Gemini 3.1 Flash Lite	0.25	1.50	26 414 107	395 735	2.5	6.9
MiniMax M2.5	0.20	1.17	125 892 755	2 456 850	9.9	39.7
GPT-5.4 Nano	0.20	1.25	39 652 301	984 189	4.0	5.9
Qwen 3.5 Plus	0.26	1.56	122 918 896	1 624 728	6.6	28.8

For each LLM, we used the default settings for inference. Nevertheless, MiniMax and Qwen invoked the SPARQL too up to 6x as many times during question answering than Gemini and GPT. The SysML domain turned out to be more challenging, requiring more SPARQL tool calls than the BPMN domain.

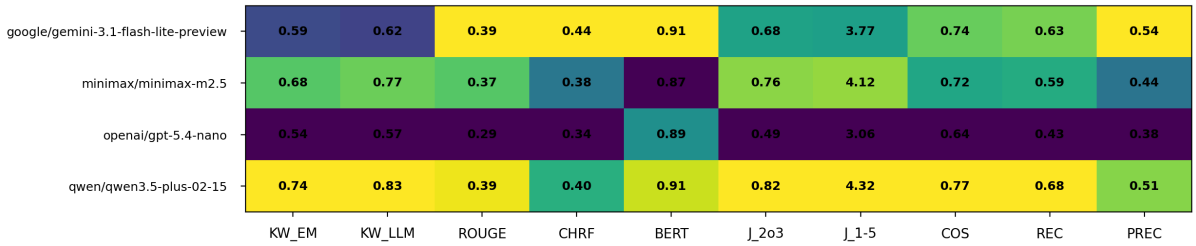
Fig. 6 shows the obtained average metric values across questions in a given application domain.

Among the selected LLMs, *Qwen 3.5 Plus performed the best in most metrics*, with MiniMax M2.5 coming close in LLM judge-based metrics **J_2o3** and **J_1-5** and Gemini 3.1 Flash light in n-gram based metrics **ROUGE** and **CHRF**, as well as **BERT**. The distribution of metrics is similar between the two domains, but model-based **PREC** and **REC** metrics are much lower for SysML than BPMN, indicating that it is more difficult find the appropriate reference elements in the TMT model.

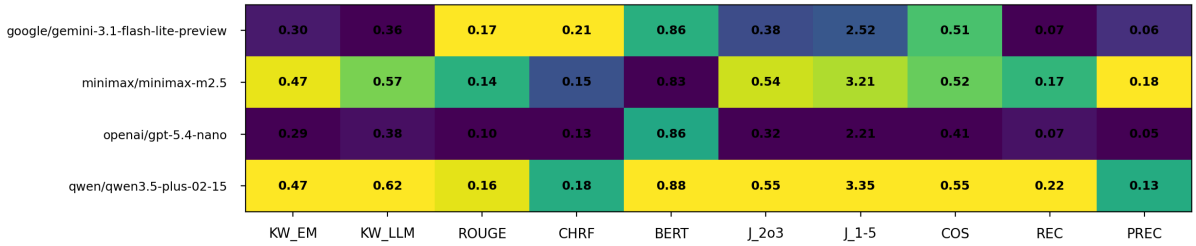
5.3. RQ2: Correlations of metrics

Fig. 7 shows a *scatterplot matrix* of metric values for different LLMs, domain, and augmentations scopes. The diagonal elements of the matrix contain the names of the metrics corresponding to the indicated row and column. Above the diagonal, the scatterplot shows the distribution of *average metric values* across the choices of LLMs, domains, and augmentation scopes. Below the diagonal, we compute the *Spearman rank correlation* ρ of metrics. We also show a *dendrogram* of single-link hierarchical clustering of metrics according to ρ as the inverse distance measure.

The pairs of metrics were highly correlated, with $\rho \geq 0.54$. According to the hierarchical clustering, metrics from the same family (keyword-based **KW_EM**, **KW_LLM**; n-gram based **ROUGE**, **CHRF**; LLM judge-based **J_2o3**, **J_1-5**; model-based **PREC**, **REC**) had the highest correlation. Judge-based and



(a) Average metric values in the BPMN domain



(b) Average metric values in the SysML domain

Figure 6: Average values of metrics in the two domains

keyword-based metrics behaved similarly, possibly indicating that the LLM judges based their scores on syntatic similarity rather than similarity in meaning. The closest to model-based metrics **PREC** and **REC** was the embedding-based **COS**, but the second embedding-based metric, **BERT** was the most dissimilar to any other metric. This may be caused by the token matching in **BERT**, which resulted in almost uniformly high scores with little distinctions between different LLMs.

In summary, the use of multiple implementations of keyword-based and judge-based metrics may be partially redundant. To supplement the syntactic metrics with semantic based ones, COS appeared to be more appropriate than judge-based metrics or **BERT**. *Model-based metrics **PREC** and **REC** cannot be entirely replaced with text-based metrics*, but show more similarity to **COS** than syntax-based metrics.

6. Discussion

We have presented a template for trustworthy modeling agents based on semantics technologies, as well as a partially LLM-generated large benchmark set based on two publicly available models from different domains. We evaluated instantiations of the assistant and the benchmark set with 10 state-of-the-art text and model-based metrics. The benchmark set, constructed with over 717M tokens of LLM inference, including over 75M tokens of Clause 4.6 Opus, and the harness were made available publicly.

Threats to validity. As an *internal threat to validity* benchmark set and ground truth generation involved the use of LLMs. To reduce the effect of non-deterministic LLM answers, the evaluation was run with 0 as temperature parameter except for **J_2o3**.

During question augmentation, our approach uses an LLM agent to generate unverified answers to help the modeling expert formulate the ground truth answers. If the agent has low recall capability in the domain, it might not find the relevant model element and come to the wrong conclusion that the element does not exist. To reduce this threat, we suggest using an LLM agent with high reasoning capacity for this step (GPT-5.2 and Claude 4.6 Opus in our case), and the *modeling expert* (★★☆) is suggested to filter out questions if they cannot verify the answer based on the given grounding model elements, or involving the *domain expert* (★★★) for special cases. Our benchmark set aims to support continuous benchmarking with automated evaluation. However, since automated evaluation cannot fully replace manual evaluation [13], there is still need for periodic human evaluation.

As an *external threat to validity*, the availability and quality of the semantic mapping limits the immediate applicability of the approach to new domains and the quality of answers. Nevertheless, we

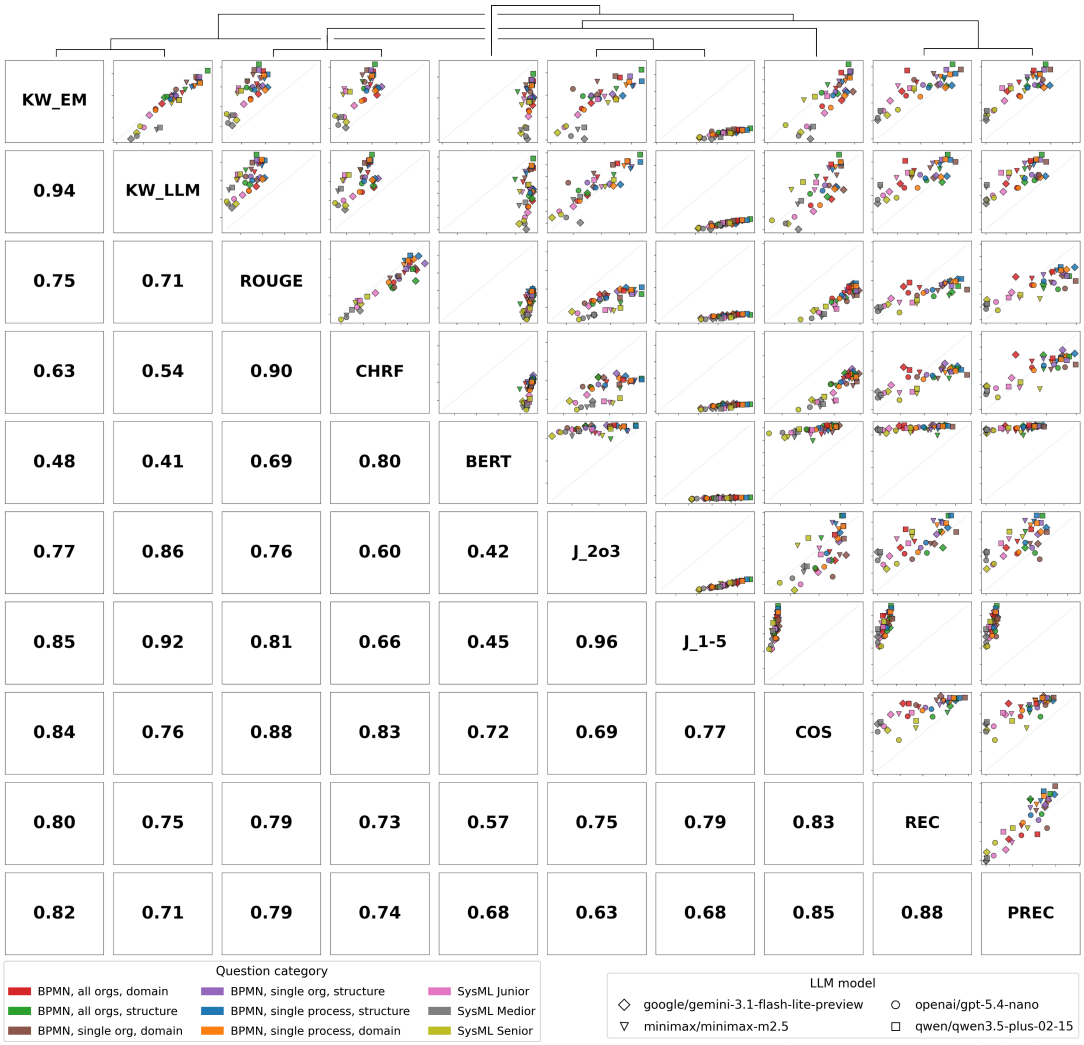


Figure 7: Scatterplot matrix of metric values

used two substantially different domains to study generalization across domains. Instantiations of the assistant with different LLMs may lead to different reasoning capabilities and answer qualities. The current evaluation focused mainly on low-cost LLMs, but the harness can be adapted to any LLM.

Future work. We plan to investigate other assistant implementations, e.g., iterative exploration of the graph schema [10] to support larger metamodels and ontologies. Moreover, RAG tools to access textual documentation or diagram images (in the case of multi-modal LLMs) could enhance the capabilities of the assistant and provide specific documentation fragments or diagrams as additional grounding.

Acknowledgments

The authors would like thank Robert Karban for the seed questions and answers for TMT. This work was partially supported by the Department of Navy award (N629092412063) issued by the Office of Naval Research, and by the European Union's Digital Europe Programme (EDGE-Skills, grant no. 101123471).

Declaration on Generative AI

During the preparation of this work, the authors used OpenAI ChatGPT 5.2-5.4 for: Drafting content, Generating literature review, and Generating code for the plots. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] TMT Observatory Corporation, Thirty Meter Telescope SysML model, 2019. URL: https://github.com/Open-MBEE/TMT-SysML-Model.

[2] N. Jankevicius, Resource analysis and automated verification for the Thirty Meter Telescope using executable SysML models, in: Workshop on Executable Modeling, 2016.

[3] S. Bachhofner, E. Kiesling, K. Revoredo, P. Waibel, A. Polleres, Automated process knowledge graph construction from BPMN models, in: DEXA, 2022. doi:10.1007/978-3-031-12423-5_3.

[4] H. Fill, P. Fettke, J. Köpke, Conceptual modeling and large language models: Impressions from first experiments with ChatGPT, Enterp. Model. Inf. Syst. Archit. (2023). doi:10.18417/EMISA.18.3.

[5] J. Cámara, J. Troya, L. Burgueño, A. Vallecillo, On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML, Softw. Syst. Model. (2023). doi:10.1007/S10270-023-01105-5.

[6] J. A. H. López, M. Földiák, D. Varró, Text2VQL: Teaching a model query language to open-source language models with ChatGPT, in: MODELS, 2024. doi:10.1145/3640310.3674091.

[7] S. Joel, J. Wu, F. Fard, A survey on LLM-based code generation for low-resource and domain-specific programming languages, ACM Trans. Softw. Eng. Methodol. (2025). doi:10.1145/3770084.

[8] Y. Yang, B. Chen, K. Chen, G. Mussbacher, D. Varró, Multi-step iterative automated domain modeling with large language models, in: MODELS-C, 2024. doi:10.1145/3652620.3687807.

[9] L. Yu, E. Alégroth, P. Chatzipetrou, T. Gorschek, Evaluating the quality of GenAI applications in software engineering: a multi-case study, Empir. Softw. Eng. (2026). doi:10.1007/S10664-025-10759-2.

[10] C. Civili, E. Sherkhonov, R. E. K. Stirewalt, Using off-the-shelf LLMs to query enterprise data by progressively revealing ontologies, in: ESWC Satellite Events, 2025. doi:10.1007/978-3-031-99554-5_46.

[11] E. Karagoz, O. J. P. Fischer, D. Mavris, Identification of missing knowledge in MBSE system models using graph-based machine learning, Syst. Eng. (2026). doi:10.1002/sys.70013.

[12] A. Dunn, sysml-bench: Evaluating tool-augmented LLMs on SysML v2 model comprehension, Preprint (2026). URL: https://gitlab.com/nomograph/sysml-bench.

[13] C. L. A. Clarke, L. Dietz, LLM-based relevance assessment still can't replace human relevance assessment, 2024. doi:10.48550/arXiv.2412.17156.

[14] S. Segura, G. Fraser, A. B. Sánchez, A. R. Cortés, A survey on metamorphic testing, IEEE Trans. Software Eng. (2016). doi:10.1109/TSE.2016.2532875.

[15] D. Sola, C. Warmuth, B. Schäfer, P. Badakhshan, J. Rehse, T. Kampik, SAP Signavio Academic Models: A large process model dataset, in: ICPM Workshops, 2022. doi:10.1007/978-3-031-27815-0_33.

[16] F. Stiehle, H. Weytjens, I. Weber, On LLM-assisted generation of smart contracts from business processes, in: DLT4BPM, 2025. doi:10.48550/arXiv.2507.23087.

[17] M. Popovic, chrF: character n-gram F-score for automatic MT evaluation, in: WMT@EMNLP, 2015. doi:10.18653/V1/W15-3049.

[18] C.-Y. Lin, ROUGE: A package for automatic evaluation of summaries, in: Text summarization branches out, 2004, pp. 74–81.

[19] S. Gupta, U. Thakar, S. Tokekar, A comprehensive survey on techniques for numerical similarity measurement, Expert Syst. Appl. (2025). doi:10.1016/j.eswa.2025.127235.

[20] N. Reimers, I. Gurevych, Sentence-BERT: Sentence embeddings using Siamese BERT-networks, in: EMNLP-IJCNLP, 2019. doi:10.18653/v1/D19-1410.

[21] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, Y. Artzi, BERTScore: Evaluating text generation with BERT, in: International Conference on Learning Representations, 2020.

[22] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT, 2019. doi:10.18653/V1/N19-1423.

[23] L. Zheng, et al., Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena, in: NeurIPS, 2023.