Evaluating LLM-Generated Conceptual Models: A Theoretical Approach for Formalizing the Calculation of Metrics on the Meta Level

Iva Vasic[1][0000−0003−4081−0203], Benedikt Reitemeyer[1][0009−0001−4131−1716], and Hans-Georg Fill[1][0000−0001−5076−5341]

University of Fribourg, Research Group Digitalization and Information Systems, 1700 Fribourg, Switzerland
[iva.vasic|benedikt.reitemeyer|hans-georg.fill]@unifr.ch
http://www.unifr.ch/inf/digits

Abstract. The generation of conceptual models using large language models (LLMs) for supporting information systems engineering is today being widely discussed. However, since each conceptual model is expressed in a specific modeling language, the evaluation approaches are often domain-dependent. In addition, previous approaches often do not fully disclose how the evaluated model elements are obtained before calculating the metrics. In this paper we thus propose a formal framework for calculating evaluation metrics on a meta level. The goal is to provide a first solid basis for the automated and thus scalable evaluation of AI-generated models. We revert to the FDMM formalism to ensure a sound formalization for modeling languages and their instantiation. We then employ two key steps in our evaluation procedure: (i) mapping of the baseline and LLM-generated models to the generic FDMM constructs, (ii) evaluation of LLM-generated models against a baseline by comparing their FDMM representations. This allows for the consistent calculation of evaluation metrics for various modeling languages using common metrics such as precision, recall, and $F1$ score. We demonstrate the application of the framework by applying the formalization to enterprise architecture models with ArchiMate and knowledge graphs in the cultural heritage domain. Finally, we compare the framework against other methodological approaches and highlight points for further extensions.

Keywords: Conceptual Model · Large Language Model · Evaluation.

1 Introduction

In the field of information systems engineering it is largely undisputed that conceptual models play an important role for the development of software and data-intensive systems [31,29]. This stretches today from the support of human understanding and communication [29,32], to the automated processing using algorithms, e.g. in model-driven engineering [8], for analytics and simulations, or via formal reasoning [6,17]. Thereby, the creation of models has been traditionally accomplished manually with the help of software-based modeling tools.

Due to the large effort required for creating models, several proposals have been made to apply natural language processing (NLP) techniques to this task in order to automate the generation of models from textual descriptions, e.g. [2,36,1]. Most recently, the availability of large language models (LLMs) has considerably boosted the quality of such generations and the effects of this technology are being increasingly studied [18,42,19].

However, there are still several open issues regarding the use of AI for modeling, one major challenge being the quality of AI output. The quality of models thereby includes physical, empirical, syntactic, semantic, pragmatic, social and deontic quality [26]. According to Krogstie, physical quality pertains to the availability of a model, i.e. that it is persistent and available for further processing. Syntactic quality relates to *syntactic completeness*, i.e. that only constructs of the syntax of the language are used, and *semantic quality* relates to the valid and complete correspondence between a model and its domain. In previous approaches, it was common to compare AI output to human-curated baselines or to some ground truth [42], whereby mostly syntactic and semantic quality were considered [12]. Similar to previous approaches in model extraction through natural language processing [4], also LLM-based model creation approaches are evaluated using metrics from information retrieval such as precision, recall, and F1 scores [42].

The evaluation of the correctness of generated models is a non-trivial task and would typically require a human expert to assess potential differences in naming and structure that are still considered correct. However, in order to automate the evaluation of generated models as required for large scale experiments, an exact, machine-processable evaluation procedure is necessary. Previously, this has been approached for example using simple matching, the use of semantic schemata or even LLMs themselves [12,43]. In addition, evaluations were specifically adapted to single modeling languages, such as class diagrams, use case diagrams, process or entity-relationship models, cf. [37].

For providing a formal basis for such evaluations that is applicable to multiple modeling languages, we thus propose in the following a generic, formal evaluation framework for arbitrary types of conceptual models. The goal of this framework is to enable the automated and thus scalable evaluation of LLM outputs using common metrics to ensure evaluations beyond human judgments. In this paper we will therefore focus first on *physical* and *syntactic* quality as *pre-requisites for other aspects of quality such as empirical, semantic and pragmatic quality* [26]. In addition we will consider attribute similarity for checking the existence of correct attribute values, including labels and reference attributes. We do not consider at this stage aspects of behavioral semantics nor semantic similarity of attributes but rather aim at the syntactic comparison to some ground truth.

For guiding the development of the framework we derived three research questions:

- **RQ1**: How can conceptual models based on different modeling languages be formally represented in a unified way for enabling the application of standard metrics for comparisons?

– **RQ2**: How can such formally represented conceptual models that are either generated by AI or by humans be quantitatively compared to baseline representations using an algorithmic approach?
– **RQ3**: What are the benefits of such an approach in comparison to previous approaches?

As shown in Figure 1, the approach is comprised of four main parts: the generation of models using CMAG prompting [20] and with the resulting output OUT_{llm}; the filtering of this output through the validation of the syntax and removal of incorrect outputs; the mapping to the formal framework based on the FDMM formalism, which includes a meta model and instance model mapping of the filtered output and the baseline model OUT_{base}; and finally the evaluation of the mapped outputs.

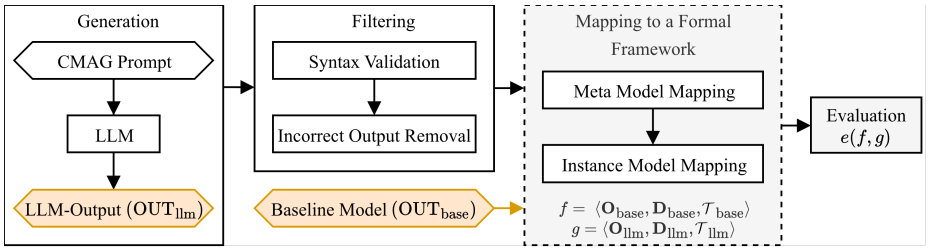

Fig. 1. Overview of our formal framework for evaluating LLM-generated conceptual models.

The remainder of the paper is structured as follows: Section 2 will outline related approaches for the evaluation of LLM-generated models. Section 3 will introduce the foundations necessary for our approach in terms of the FDMM formalism and the CMAG prompt structure. Section 4 presents our framework, followed by two application examples in Section 5 that exemplify the framework usage for formalizing enterprise architecture models and knowledge graphs. Section 6 discusses the approach and its comparison to previous approaches. Section 7 concludes the paper with an outlook on future work.

2 Related Work on Evaluating LLM-generated Models

Since LLMs demonstrate impressive capabilities in generating conceptual models, evaluation approaches have likewise evolved. From the existing literature on evaluating generated models, frequently identified methods include the manual verification by experts, the comparison of the LLM output to some ground truth dataset, and evaluation using LLMs themselves.

As LLM-generated output has not yet reached fully reliable outcomes, human experts remain essential in the evaluation process. For software modeling with UML class diagrams, Shehata et al. [40] evaluate GPT-generated models

by involving experts who compare the models to ground truth diagrams. Automated domain modeling with LLMs has also been tested on UML class diagram generation and evaluated by experts in [11], highlighting the challenge to reach consensus among human modelers. In the work of De Bari et al. [14], both students and LLMs were employed to generate UML diagrams based on several modeling exercises, whose solutions were then compared and checked against syntactic, semantic, and pragmatic errors by a human expert. Cámara et al. [13] observed various flaws in ChatGPT-generated software models by evaluating the output themselves manually.

In the work of [35], LLM-generated class, attribute, and association generators are evaluated by manually assigning matching scores across various strictness criteria, i.e., beyond strict matching like in our workflow, after which the statistical analysis was reported for each level of strictness. In a similar idea to ours, that is, primarily mapping to a formal framework before evaluation, Lo et al. [28] leveraged knowledge graphs to map ontologies and measure their similarity with the ground truth. They tested semantic and structural similarity in a fuzzy manner, which gives different results than strict matching, by using edges and subgraphs as evaluation elements.

As LLMs advance, they have emerged as tools for evaluation themselves [45], as presented by Calamo et al. [10] for assessing generated UML class diagrams against the human-generated dataset. Huang et al. [23] used a fine-tuned LLM to evaluate the factual correctness of knowledge graph triples. Precision, recall, and $F1$ score metrics were employed by Silva et al. [41] to evaluate the Tree-of-Thoughts prompting technique for UML class diagram generation, comparing it against a reference solution using an LLM-based evaluation. However, Ibáñez et al. [24] found that LLMs are unreliable in evaluating UML class diagram designs compared to expert evaluation performance, due to the inability to interpret semantics and evaluative reasoning. On the other hand, the findings of Tsaneva et al. [43] indicate that the inclusion of LLMs in the knowledge graph evaluation process increases the precision while negatively affecting the recall and consequently the $F1$ score. Moreover, the hybrid approach, where human involvement complements the LLM evaluator, demonstrated the best overall outcome.

Despite these previous efforts, a formal framework for evaluations against a baseline of models of arbitrary type has not yet been proposed. We thus present in the following a solution approach based on the FDMM formalism for laying the basis for automated and scalable evaluations.

3 Foundations on Formalizing Conceptual Models

In this chapter we will describe the foundations required for our approach. This will include a brief characterization of the core parts of FDMM and the CMAG prompt structure. FDMM (Formalism for Describing Metamodels and Models) was developed to provide an exact description of meta models and their instantiation to models using only set theory and first-order logic [21]. Its original intention was to formalize modeling languages in ADOxx, which is widely

used in domain-specific conceptual modeling. It has been applied e.g. to process modeling [39,25], service modeling [5], or rule modeling [34]. For the scope of this paper we will use it for describing meta models and models on a common, technology-independent level.

According to FDMM, meta models **MM** are used for deriving model instances **mt**. A meta-model is a tuple $\mathbf{MM} = \langle \mathbf{MT}, \preceq, \text{domain}, \text{range}, \text{card} \rangle$ where **MT** is the set of model types and $\mathbf{MT}_i$ is a tuple $\mathbf{MT}_i = \langle \mathbf{O}_i^T, \mathbf{D}_i^T, \mathbf{A}_i \rangle$, which consists of a set of object types $\mathbf{O}_i^T$, a set of data types $\mathbf{D}_i^T$, and a set of attributes $\mathbf{A}_i$.

The attributes $\mathbf{A}_i$ will also be used for expressing relations between object types through references. This will be used for realizing attributes that point to other object types as well as for describing the start and end of edges, which are described as object types as well. The object types, data types, and attributes are part of their respective total sets, i.e. $\mathbf{O}^T = \bigcup_j \mathbf{O}_j^T, \mathbf{D}^T = \bigcup \mathbf{D}_i^T, \mathbf{A} = \bigcup \mathbf{A}_i$. The $\preceq$ operator describes an ordering on the set of object types $\mathbf{O}^T$, in the sense that $o_1^t \preceq o_2^t$ states that the object type o_1^t is a subtype of the object type o_2^t and inherits its attributes.

FDMM defines functions for attaching attributes to object types, for defining the range of attribute values, and for the number of attribute values per object type instance. The domain function maps attributes to the power set of all object types, i.e. $\text{domain} : \mathbf{A} \to \mathcal{P}(\bigcup_j \mathbf{O}_j^T)$. It will constrain what objects an attribute can map in the model instances and is thus used to attach attributes to a particular set of object types. The range function maps an attribute to the power set of all pairs of object types and model types, all data types, and all model types, i.e. $\text{range} : \mathbf{A} \to \mathcal{P}\left(\bigcup_j (\mathbf{O}_j^T \times \{\mathbf{MT}_j\}) \cup \mathbf{D}^T \cup \mathbf{MT}\right)$. On the level of the model instances, the range function will later constrain what values an attribute can have. For the definition of a meta model it serves to describe the type of an attribute. We can distinguish three cases that may be combined: *i.* where the attribute points to an instance of an object type that is contained in a particular model type $(\mathbf{O}_j^T \times \{\mathbf{MT}_j\}$; *ii.* where the attribute points to an instance of a datatype $\mathbf{D}^T$, or *iii.* where the attribute points to an instance of a model type **MT**. With the card function it can be specified how many instances of object types, data types, or model types an attribute of a particular object type may have, i.e. $\text{card} : O^T \times A \to \mathcal{P}(\mathbb{N} \times (\mathbb{N} \cup \{\infty\}))$. FDMM defines several correctness criteria for meta models, which we will omit here for brevity and which the interested reader may find in the original source [21].

Next, we describe how FDMM defines the instantiation of meta models to models. For this purpose, FDMM maps model types, object types, and data types to model instances, objects, and data values. In addition, so-called *triples* are defined that assign values of attributes to objects. The instantiation of a metamodel **MM** is a tuple $\langle \mu_{\mathbf{mt}}, \mu_{\mathbf{O}}, \mu_{\mathbf{D}}, \mathcal{T}, \beta \rangle$. $\mu_{\mathbf{mt}}$, $\mu_{\mathbf{O}}$, and $\mu_{\mathbf{D}}$ are functions that map from the elements in the meta model to instances.

The function $\mu_{\mathbf{mt}} : \mathbf{MT} \to \mathcal{P}(\mathbf{mt})$ maps model types to the power set of model instances. The function $\mu_{\mathbf{O}} : \bigcup_j \left(\mathbf{O}_j^T \times \{\mathbf{MT}_j\}\right) \to \mathcal{P}(\mathbf{O})$ stands for the mapping of object types of a specific model type to the power set of object

instances where the set of all object instances is $\mathbf{O} = \bigcup_j \mu_{\mathbf{O}}(\mathbf{O}_j^T \times \{\mathbf{MT}_j\})$. For brevity we omit here the specification of abstract object type, which can be specified in FDMM and that is not to be instantiated directly. The function $\mu_{\mathbf{D}} : \mathbf{D}^T \rightarrow \mathcal{P}(\mathbf{D})$ maps data types to the power set of data objects. Further constraints on data types are not contained in FDMM itself. Rather, it is left to the user to add them if necessary or otherwise refer to commonly known data types such as integer numbers or strings.

Finally, triples $\mathcal{T} \subseteq \mathbf{O} \times \mathbf{A} \times (\mathbf{D} \cup \mathbf{O} \cup \mathbf{mt})$ are used to describe the contents of model instances. They combine object instances and their attributes with instances of data types, other object instances, or model instances. The triples are then assigned to concrete model instances via the map $\beta : \mathbf{mt} \rightarrow \mathcal{P}(\mathcal{T})$.

3.1 CMAG Prompt Structure

The CMAG (Conceptual Model Augmented Generative Artificial Intelligence) framework positions conceptual models as interfaces to large language models during the prompting and the generation phase [20]. It proposes a prompt structure based on the following elements: a *few shot context* for informing the LLM about the modeling language, a *task framing* to reference the modeling language, the actual *task content* for describing the intended answer of the LLM, and the *output description* that refers again to the modeling language. As LLMs can adapt their responses through few-shot prompting, they may be guided with information about a meta[2] model, metamodel and sample instances in the few shot context. Alternatively, the task framing and the output description may reference a modeling language that the LLM has already been trained on. The result of this prompt structure is that LLMs can be forced to create models for a given task content. For example, instead of producing the description of the steps of a business process for some service in textual form, the LLM can be thus forced to create a model in BPMN notation. Similarly, other schemata for modeling languages can be defined specifically for CMAG prompts or existing languages re-used. As a consequence, users can more easily assess an LLM's output and conduct modifications in the created models, which can be sent back to the LLM for further processing, e.g. in agentic workflows [16].

Thus, the expected output from an LLM after using the CMAG prompt is a model instance ($\mathrm{OUT}_{\mathrm{llm}}$) that tends to exhibit higher accuracy than the output obtained from an unstructured regular prompt to an LLM.

4 Formal Evaluation Framework Based on FDMM

Following our schematic workflow in Figure 1, we advance now with the presentation of the two main blocks for the evaluation after the generation of $\mathrm{OUT}_{\mathrm{llm}}$: the filtering of the raw LLM-generated output, the mapping to FDMM and the formal definition of evaluation metrics.

4.1 Filtering LLM-generated Output

Considering that the LLM output is inherently unstructured and represented in a raw format, its filtering becomes an important pre-processing step to discard undesired outputs before the formal evaluation. Hence, we perform a strict filtering on syntactically incorrect instances. More precisely, if any part of the generated output is syntactically incorrect, the entire output instance is discarded. The filtering is thereby accomplished using modeling language specific validation standards since there is no universal tool for evaluating the syntactical accuracy of arbitrary model instances. For instance, we used an RDF validator for RDF instances, which focuses on structural correctness, whereas the validator we use for ArchiMate models additionally performs some semantic checks. This can be considered a limitation of the approach since it needs to be adapted to each language individually once. However, once the syntactically erroneous instances are discarded, the workflow can proceed to mapping LLM-generated output $\mathrm{OUT}_{\mathrm{llm}}$ to FDMM.

4.2 Mapping to FDMM and Formal Definition of Evaluation Metrics

Our assumption is that every conceptual model can be mapped to FDMM constructs, as the formalism provides a sufficiently general framework to represent diverse domain-specific languages. The mapping of a modeling language to FDMM needs to be defined once and can subsequently be used for an arbitrary number of model instances based on this language, thereby ensuring the scalability of the approach.

A formal evaluation step is defined as $e(f, g)$, where f is a result of mapping any baseline model ($\mathrm{OUT}_{\mathrm{base}}$) to FDMM, such that $f = \langle \mathbf{O}_{\mathrm{base}}, \mathbf{D}_{\mathrm{base}}, \mathcal{T}_{\mathrm{base}} \rangle$, and g is a result of mapping any LLM output ($\mathrm{OUT}_{\mathrm{llm}}$) to FDMM, such that $g = \langle \mathbf{O}_{\mathrm{llm}}, \mathbf{D}_{\mathrm{llm}}, \mathcal{T}_{\mathrm{llm}} \rangle$. The general function e represents any evaluation metric instantiated by the list of specific measures, such as lexical and semantic included in [9]. In this study, only a few of them are used for representation. We use subscripts "base" and "llm" to denote the baseline and LLM-generated elements. Considering any meta model $\mathbf{MM}$ and its model types $\mathbf{MT}_i = \langle \mathbf{O}_i^T, \mathbf{D}_i^T, \mathbf{A}_i \rangle$, the evaluation sets — LLM-generated object type instances $\mathbf{O}_{\mathrm{llm}}$, data type instances $\mathbf{D}_{\mathrm{llm}}$, and triples $\mathcal{T}_{\mathrm{llm}}$ — are being compared to their respective baseline counterparts $\mathbf{O}_{\mathrm{base}}$, $\mathbf{D}_{\mathrm{base}}$, and $\mathcal{T}_{\mathrm{base}}$.

Next, we introduce an FDMM-based formal foundation of the evaluation framework, which serves for calculating metrics commonly used in the literature. We define $S_{\mathbf{O}}$, $S_{\mathbf{D}}$, and $S_{\mathcal{T}}$, each providing insights into whether the expected FDMM elements are generated. Given the baseline and LLM-generated sets of elements, we can write a simple matching score expression: $S_{\mathbf{O}} = |\mathbf{O}_{\mathrm{base}} \cap \mathbf{O}_{\mathrm{llm}}|$, $S_{\mathbf{D}} = |\mathbf{D}_{\mathrm{base}} \cap \mathbf{D}_{\mathrm{llm}}|$, $S_{\mathcal{T}} = |\mathcal{T}_{\mathrm{base}} \cap \mathcal{T}_{\mathrm{llm}}|$.

However, the comparison is not always straightforward since LLMs occasionally generate additional, unexpected elements. Furthermore, the quantity of generated elements might fall below that in the baseline sets, additionally prompting

for extending the evaluation strategy. Therefore, we introduced a new evaluation criterion, which respectively shows whether there exist additional or missing set elements in the LLM-generated output. Accordingly, symbols $(+)$, $(-)$, and Δ are assigned to the score S to respectively denote missing elements, extra generated elements, and their difference. We denote a general scoring function $S_{(\cdot)}$ where the subscript denotes one of the sets $\mathbf{O}$, $\mathbf{D}$, or $\mathcal{T}$. We can now generalize the expressions for $S_{\mathbf{O}}$, $S_{\mathbf{D}}$, and $\mathcal{T}$ into $S_{(\cdot)} = |(\cdot)_{\text{base}} \cap (\cdot)_{\text{llm}}|$, where $(\cdot) \in \{\mathbf{O}, \mathbf{D}, \mathcal{T}\}$. Similarly, for any baseline and LLM-generated sets, and considering the defined notation $(\cdot)$ previously, we introduce: $S^{-}_{(\cdot)} = |(\cdot)_{\text{base}} \setminus (\cdot)_{\text{llm}}|$, $S^{+}_{(\cdot)} = |(\cdot)_{\text{llm}} \setminus (\cdot)_{\text{base}}|$, and $S^{\Delta}_{(\cdot)} = S^{+}_{(\cdot)} - S^{-}_{(\cdot)}$, where $S^{-}_{(\cdot)}$ denotes the number of missing elements in LLM-generated sets and $S^{+}_{(\cdot)}$ denotes the number of extra elements in the LLM-generated sets and $S^{\Delta}_{(\cdot)}$ refers to the final difference between missing and extra counts in the generated sets. The sign of $S^{\Delta}_{(\cdot)}$ indicates the balance between extra and missing elements, such that $S^{\Delta}_{(\cdot)} > 0$, $S^{\Delta}_{(\cdot)} = 0$, $S^{\Delta}_{(\cdot)} < 0$, if it holds $S^{+} > S^{-}$, $S^{+} = S^{-}$, $S^{+} < S^{-}$, respectively.

Guided by the evaluation metrics in the literature, our approach enables the formal formulation of standard metrics for models of any type[1], such as precision, recall, and F_1-score. For simplicity, our metric considers strict matching rules although it could be extended to the semantic precision and recall similarly to proposed research by Euzenat [15]. The evaluation function $e(f, g)$ introduced earlier is formally expressed in a general way as follows:

$$p_{(\cdot)} = \frac{S_{(\cdot)}}{|(\cdot)_{\text{llm}}|}, \quad r_{(\cdot)} = \frac{S_{(\cdot)}}{|(\cdot)_{\text{base}}|}, \quad F_{1(\cdot)} = \frac{2p_{(\cdot)}r_{(\cdot)}}{p_{(\cdot)} + r_{(\cdot)}} \quad ; \tag{1}$$

where p and r stand for precision and recall. For a collection of K evaluated models, indexed by $k = 1, \ldots, K$, the aggregated statistics of these metrics (Equation 1) can be expressed through the mean and standard deviation. The empirical evaluation that can accordingly be derived is presented in the following section.

5 Demonstration of Application

For demonstrating the application of our approach, we apply the mapping step to FDMM to two modeling languages from very separate domains. First to the ArchiMate modeling language and second to knowledge graphs[2].

[1] Note that recall and F_1-score are calculated in terms of the baseline set instance $(\cdot)_{\text{base}}$ rather than the world set relevant elements. The inclusion of the world set of relevant values would demand extending the baseline in order to further classify the extra LLM-generated instance. This is envisaged for future work.

[2] Code and supplementary material can be found online at: https://doi.org/10.5281/zenodo.21263688

5.1 Enterprise Architecture Models in ArchiMate

ArchiMate [33] is an open and independent modeling language for enterprise architecture that provides instruments to describe, analyze, and visualize relationships among business domains used by many large organizations [27]. The elaborations in the following are based on Open Group's ArchiMate version 3.2 [33] and the concrete realization of ArchiMate in PlantUML, which we found to work well in LLM-based settings.

As can be seen in Figure 2, mapping PlantUML and ArchiMate to FDMM is accomplished through the definition of a meta model composed of one model type $\mathbf{MM} = \{\mathbf{MT}_{\mathrm{AM}}\}$ where $\mathbf{MT}_{\mathrm{AM}} = \langle \mathbf{O}^T_{\mathrm{AM}}, \mathbf{D}^T_{\mathrm{AM}}, \mathbf{A}_{\mathrm{AM}} \rangle$. Then we assume $\mathbf{O}^T_{\mathrm{AM}} = \{\mathrm{Category},\ \mathrm{ElementType},\ \mathrm{Element}, \mathrm{RelationType},\ \mathrm{Relation}\}$, $\mathbf{D}^T_{\mathrm{AM}} = \{\mathrm{string}\}$, and finally $\mathbf{A}_{\mathrm{AM}} = \{\mathrm{relationName},\ \mathrm{description},\ \mathrm{hasRelationType},\ \mathrm{hasCategory},\ \mathrm{hasElementType},\ \mathrm{identifier},\ \mathrm{label},\ \mathrm{fromElement}, \mathrm{toElement},\ \mathrm{elementName},\ \mathrm{fromElementType},\ \mathrm{toElementType},\ \mathrm{categorization},\ \mathrm{categoryName}\}$.

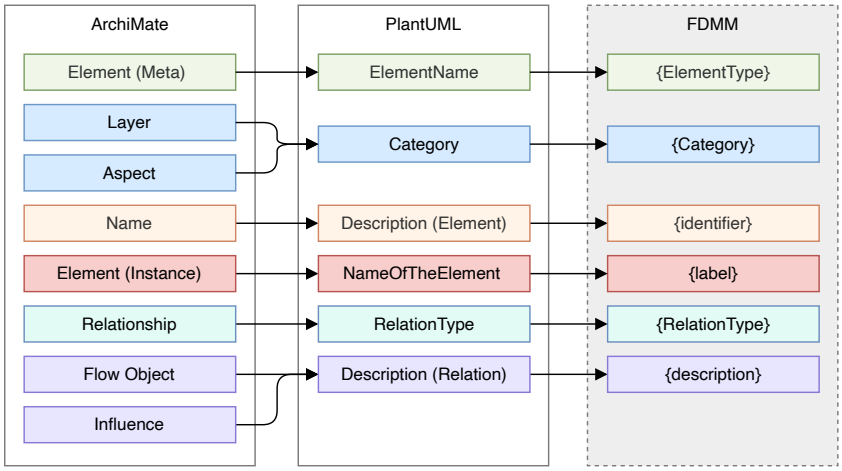

Fig. 2. Mapping between ArchiMate and PlantUML for formalization in FDMM

A concrete model can then be instantiated as $\mu_{\mathbf{mt}}(\mathbf{MT}_{\mathrm{AM}}) = \{\mathbf{mt}_{\mathrm{am}_1}\}$ along with some object types instances $\mu_{\mathbf{O}}(\mathrm{Category}, \mathbf{MT}_{\mathrm{AM}}) = \{Business\}$, $\mu_{\mathbf{O}}(\mathrm{Element}, \mathbf{MT}_{\mathrm{AM}}) = \{Underwrite\}$, $\mu_{\mathbf{O}}(\mathrm{ElemenType}, \mathbf{MT}_{\mathrm{am}}) = \{Process\}$ and data type instances $\mu_{\mathbf{D}}(\mathrm{string}) = \{underwrite, UnderwriteRisk\}$. Finally, we describe triples of the model instance, $(Process$ categorization $Business) \in \beta(\mathbf{mt}_{\mathrm{am}_1})$ or $(Underwrite$ label "$UnderwriteRisk$"$) \in \beta(\mathbf{mt}_{\mathrm{am}_1})$.

The filtering step was executed using the PlantUML web editor. In the subsequent phase, only models that were deemed valid were transferred to FDMM. The majority of invalidations were attributed to prefixes and suffixes surrounding the PlantUML code or the inclusion of non-existent elements, such as "Business

Domain". Through this mapping we can show that ArchiMate models in PlantUML notation can be successfully mapped to FDMM, thus providing the basis for calculating the comparison metrics.

5.2 Knowledge Graphs in Cultural Heritage

The second demonstration refers to the generation of knowledge graphs in the domain of cultural heritage, similar to the work of Vasic et al. [44]. Knowledge graphs represent formal structures of information systems where entities (also known as nodes and vertices) and semantic relationships are connected as a graph [22]. Two entities and their mutual relationships form a triple, often represented in RDF format as subject, predicate, and object [30]. The Getty Collection was chosen as a reference dataset in this study, and it is organized through the Linked Art model [38] built upon Linked Open Data standards and serialized in RDF-compatible JSON-LD format.

Since certain parts of the Getty Collection URIs are entirely specific to the internal organization of the museum, we intentionally do not consider them in their full format, as our objective is not to evaluate LLM output against these specific identifiers. Instead, we simplify them by introducing a custom prefix *concept* and mapping the Getty-specific one to it. For example, we introduced concept:object, concept:material, and others listed in the supplementary material. Moreover, since our generation and evaluation algorithm processes one painting at a time, there is no possibility that one concept overlaps with another contained in the rest of the artworks.

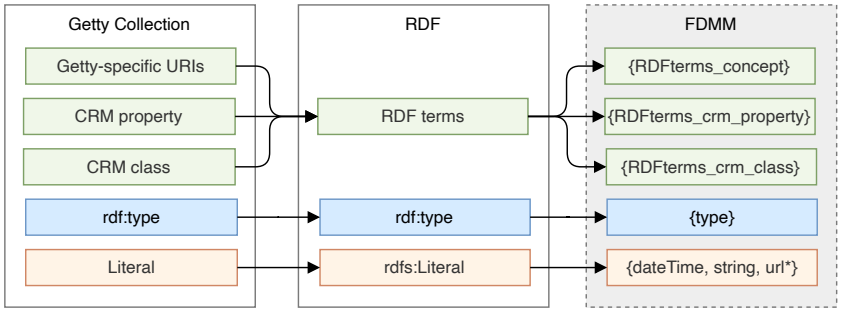

Fig. 3. Mapping Getty collection knowledge graphs elements to FDMM through RDF. (*) denotes elements that exist in the model but are excluded from the generation or the evaluation.

Next, we describe the mapping process of the baseline model $\mathrm{OUT}_{\mathrm{base}}$ and the resulting tuples $\langle \mathbf{O}_{\mathrm{base}}, \mathbf{D}_{\mathrm{base}}, \mathcal{T}_{\mathrm{base}} \rangle$. A meta model is defined involving one model type, $\mathbf{MM} = \{\mathbf{MT}_{\mathrm{GC}}\}$, for which the subscript is assigned according to the Getty Collection and CIDOC-CRM ontology, and where $\mathbf{MT}_{\mathrm{GC}} =$

$\langle \mathbf{O}^T_{\mathrm{GC}}, \mathbf{D}^T_{\mathrm{GC}}, \mathbf{A}_{\mathrm{GC}} \rangle$. Then we define object types, data types, and attributes, respectively, as follows: $\mathbf{O}^T_{\mathrm{GC}}$ = {RDFterms_concept, RDFterms_crm_property, RDFterms_crm_class}, $\mathbf{D}^T_{\mathrm{GC}}$ = {dateType, dateTime, url}, and $\mathbf{A}_{\mathrm{GC}}$ = {type, has_concept, has_property, literal, identifier}. Note that we introduced *identifier* as an attribute construct that links the RDFterms_concept to the original Getty identifiers[3]. Its role serves to complete our mapping but is excluded from evaluation, as it is meant to map back the previously mentioned *concepts* to the Getty URI. Figure 3 presents the mapping of the Getty Collection concepts to FDMM, with alignment to the RDF concepts.

In the model instantiation phase, we may assume $\mu_{\mathbf{O}}(concept, \mathbf{MT}_{\mathrm{GC}}) = \{concept : material\}$ and $\mu_{\mathbf{O}}(RDFterms\_crm\_property, \mathbf{MT}_{\mathrm{GC}}) = \{crm : P190\_has\_symbolic\_content\}$ as object type instances. We further assume one data type instance as $\mu_{\mathbf{D}}(string, \mathbf{MT}_{\mathrm{GC}}) = \{'Oil\ on\ canvas'\}$. Finally, we may write triples out of these elements to express the material used for a painting: (concept:material *has_property* crm:P190_has_symbolic_content) $\in \beta(\mathbf{mt}_{gc})$ and (crm:P190_has_symbolic_content *literal* 'Oil on canvas') $\in \beta(\mathbf{mt}_{gc})$. The filtering step was performed using an open-source tool[4].

5.3 Demonstration of the Calculation of Metrics

For demonstrating the calculation of the evaluation metrics introduced in 4.2, we show in the following only first results for generating 20 knowledge graphs for paintings in the cultural heritage domain to give an impression how results can be generated. We used Gemma3:12b, GPT-oss-120b, and Gemini-2.5-pro models. The CMAG prompt structure was used to force model creation for a given task content [20]. At first, a CMAG context reference was provided to each LLM, which points to the Getty Collection and CIDOC-CRM ontology. In the prompt, the LLM is requested to generate a model in Turtle format of triple statements for a given painting description. The included reference painting was chosen from the Getty Collection[5]. Additionally, the context reference includes the required concepts that will be evaluated after, such as the painting's place and date of creation, artist, and the materials used, among others.

The model in CMAG terms includes an example for a few-shot prompt. Finally, a pattern is included, representing a painting description, based on which the KG should be derived. The filtering step was then performed using an open-source tool[4], which reported that all LLM-generated graphs complied with the Turtle syntax. Therefore, 20 knowledge graphs, each related to a specific painting, were considered for evaluation. The summary of missing and extra LLM-generated elements, and their respective differences, are represented in Table 1

[3] An example of the original Getty identifier is `https://data.getty.edu/museum/collection/object/e79dc10c-da13-481a-8e81-4b30474d6778`.

[4] Available at: [`https://github.com/IDLabResearch/TurtleValidator`], accessed: 14.11.2025

[5] *Portrait of a Woman*, available at: [`https://www.getty.edu/art/collection/object/114T8J`], accessed 05.11.2025

Table 1. Illustrative evaluation scores for the knowledge graph use case.

LLM	$S_{\mathbf{O_{GC}}}$	$S^-_{\mathbf{O_{GC}}}$	$S^+_{\mathbf{O_{GC}}}$	$S^\Delta_{\mathbf{O_{GC}}}$	$S_{\mathbf{D_{GC}}}$	$S^-_{\mathbf{D_{GC}}}$	$S^+_{\mathbf{D_{GC}}}$	$S^\Delta_{\mathbf{D_{GC}}}$	$S_{\mathcal{T}_{\mathrm{GC}}}$	$S^-_{\mathcal{T}_{\mathrm{GC}}}$	$S^+_{\mathcal{T}_{\mathrm{GC}}}$	$S^\Delta_{\mathcal{T}_{\mathrm{GC}}}$
Gemini	440	8	17	9	**119**	9	18	9	794	26	77	51
Gemma	446	2	14	12	118	10	21	11	**808**	12	71	59
GPT	**447**	1	13	12	110	18	29	11	801	19	79	60

as well as the calculation of first results of precision, recall, and F1 score in Table 2. Note that these results are only for verifying the feasibility of the calculations due to limitations of space here. For a sound evaluation, a larger number of LLMs as well as variations on the number of runs and parameter settings would need to be included in addition. We reported the descriptive statistics across the 20 paintings, however, no statistical significance tests were performed because it is currently beyond the scope of the study to measure the differences in performance across LLMs. Therefore, with such illustrative analysis, we cannot conclude on whether the observed differences across LLMs are statistically significant.

Table 2. Illustrative statistical results for the knowledge graph use case: Precision, recall, and F_1-score are reported as mean ($\bar{\cdot}$) ± standard deviation (σ) across the K generated models ($K = 20$).

	precision (%)		
	$\mathbf{O}^T_{\mathrm{GC}}$	$\mathbf{D}^T_{\mathrm{GC}}$	$\mathcal{T}$
Gemini	96.30 ± 4.06	**87.14 ± 13.83**	91.25 ± 8.48
Gemma	96.95 ± 2.86	84.64 ± 12.75	**91.91 ± 6.08**
GPT	**97.17 ± 2.13**	79.16 ± 14.96	91.02 ± 6.19
	recall (%)		
	$\mathbf{O}^T_{\mathrm{GC}}$	$\mathbf{D}^T_{\mathrm{GC}}$	$\mathcal{T}$
Gemini	98.22 ± 3.65	**92.86 ± 12.09**	96.81 ± 5.88
Gemma	99.54 ± 1.40	92.14 ± 13.23	**98.52 ± 2.20**
GPT	**99.78 ± 0.97**	85.71 ± 13.88	97.67 ± 2.21
	F1 Score (%)		
	$\mathbf{O}^T_{\mathrm{GC}}$	$\mathbf{D}^T_{\mathrm{GC}}$	$\mathcal{T}$
Gemini	97.23 ± 3.51	**89.61 ± 12.05**	93.84 ± 6.63
Gemma	98.22 ± 1.99	88.10 ± 12.49	**95.02 ± 3.76**
GPT	**98.45 ± 1.26**	82.14 ± 14.01	94.15 ± 3.86

6 Discussion and Guidelines for Applying the Framework

With our approach we could show how an evaluation of LLM-generated conceptual models can be addressed in a modeling language agnostic way, thereby

answering RQ1 and RQ2. For answering RQ3, we present in Table 3 a comparison of our approach against previous approaches. The comparison focuses only on the evaluation step rather than the methods for the generation of conceptual models, pre-processing, or post-processing.

Table 3. Comparison of techniques and metrics used for evaluating LLM-generated conceptual models across studies in the literature. The reported values for our approach are reported in the last column. NS - Not Specified

Article	*Automatic/ Human in the Loop*	*Code Available*	*Output Format(s)*	*Evaluation Metrics*	*Modeling Language Independent*	*Evaluation Artefacts*	*Formal Representation*
[7]	A	✓	Custom	Mean, Standard Deviation, Significance, Precision, Recall, F0.5 Score, F1 Score, F2 Score	✗	Classes, Associations	✗
[40]	H	✗	PlantUML	Average structure rating	✗	Classes, Attributes, Relationships	✗
[11]	H	✓	Custom	Precision, Recall, F1 Score	✗	Classes, Attributes, Relationships	✗
[3]	NS	✓	Custom	F1 Score	✗	Custom DSL	✗
[10]	A	✓	PlantUML	Precision, Recall, F1 score	✗	Classes, Associations, Attributes (Name, Cardinality)	✗
[35]	H	✓	JSON	Precision, Recall, F1 Score	✗	Classes, Attributes, Associations	✓
[41]	A	✓	PlantUML	Precision, Recall, F1 Score	✗	Classes, Attributes, Relationships, Association Classes	✓
[44]	H	✓	JSON	Agreement rate	✗	CIDOC-CRM class Value	✗
Ours	H	✓	Turtle, PlantUML	Precision, Recall, F1 Score	✓	$\mathbf{O}^T$, $\mathbf{D}^T$, $\mathcal{T}$	✓

We distinguish between purely *Automatic* and *Human in the Loop* where at least some step is conducted manually. In the column *Modeling Language Independent* we assess whether an approach is designed as independent of a particular modeling language. Although some approaches use generic constructs, e.g. for describing domain models, this has so far not been the case.

Another advantage of our approach is that we formally express the evaluation artifacts in terms of FDMM, while most other approaches do not present this information, which may limit the applicability when evaluating other types of models. The column *Formal Representation* indicates that the evaluated el-

ements are formally represented, either in mathematical terms, through an implementation, or using models themselves.

6.1 Guidelines for Applying the Framework

For applying our framework we derived the following guidelines to ensure its correct and efficient usage. First, potential users need to familiarize themselves with the FDMM formalism and how to map modeling languages to it. Potential sources include previous contributions to this topic, e.g. [39,25,5,34]. Second, the mapping of a modeling language to FDMM requires high familiarity with the modeling language to guarantee a correct mapping. Ideally, the creators of the modeling language or senior users are involved in this step. Finally, the setup of the automated pipeline includes the choice of a suitable tool for checking the syntactical correctness of generated models. This depends on the used modeling language. It is preferable to use official validation tools for this step if available or to use at least well-documented formats such as PlantUML or Mermaid that can be well processed by current AI models.

6.2 Limitations

In contrast to other approaches, our approach is the only one that has been applied to multiple modeling languages. However, this comes at the cost of having to map first the language constructs to FDMM. At the moment this has to be performed manually, however, it does not affect scalability in terms of the number of evaluated model instances later. Further, our approach so far mainly considers the syntactic quality of models. However, we consider it as a sound basis for adding the evaluation of semantics in the future.

7 Conclusion

This paper introduced a formal framework based on the FDMM formalism for evaluating LLM-generated models on a meta level, by formally describing the contents of metamodels and their instantiation to models on the level of syntax. The evaluation was presented as a function of comparison of LLM-generated and baseline model tuples formed from FDMM constructs. The strength of our approach lies in its ability to generalize across different domains, compared to other approaches which, to our knowledge, focus on a single domain.

One limitation of our approach stems from the manual effort involved in the mapping process and from relying primarily on syntactical similarity in the comparison phase. Future work will extend the approach towards including additional semantic measures such as reproducible semantic evaluations and providing further support, e.g. for automating the mapping to FDMM.

8 Acknowledgment

This work was financially supported by the Smart Living Lab, a joint project funded by the University of Fribourg, EPFL, and HEIA-FR.

References

1. van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L.: Challenges and Opportunities of Applying Natural Language Processing in Business Process Management. In: International Conference on Computational Linguistics. pp. 2791–2801. Association for Computational Linguistics (2018)
2. Arora, C., Sabetzadeh, M., Briand, L., Zimmer, F.: Extracting domain models from natural-language requirements: approach and industrial evaluation. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems. pp. 250–260. MODELS '16, ACM, New York, NY, USA (2016)
3. Arulmohan, S., Meurs, M.J., Mosser, S.: Extracting Domain Models from Textual Requirements in the Era of Large Language Models. In: 2023 ACM/IEEE MODELS-C. pp. 580–587 (Oct 2023)
4. Bellan, P., Dragoni, M., Ghidini, C., Aa, H.v.d., Ponzetto, S.P.: Process Extraction from Text: Benchmarking the State of the Art and Paving the Way for Future Challenges. arXiv (Oct 2023). https://doi.org/10.48550/arXiv.2110.03754
5. Boucher, X., Medini, K., Fill, H.G.: Product-Service-System Modeling Method. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 455–482. Springer (2016)
6. Braga, B.F.B., Almeida, J.P.A., Guizzardi, G., Benevides, A.B.: Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. Innovations in Systems and Software Engineering **6**(1), 55–63 (Mar 2010)
7. Bragilovski, M., van Can, A.T., Dalpiaz, F., Sturm, A.: Leveraging machines to derive domain models from user stories. Requirements Engineering **30**(2), 241–262 (2025)
8. Brambilla, M., Cabot, J., Wimmer, M.: Model-driven software engineering in practice, second edition. Synthesis Lectures on Software Engineering **3**(1), 1–207 (2017)
9. Bulygin, L.: Combining Lexical and Semantic Similarity Measures with Machine Learning Approach for Ontology Matching Problem. In: Kalinichenko, L.A., Manolopoulos, Y., Stupnikov, S.A., Skvortsov, N.A., Sukhomlin, V. (eds.) Selected Papers of the XX International Conference on Data Analytics and Management in Data Intensive Domains (DAMDID/RCDL 2018), Moscow, Russia, October 9-12, 2018. CEUR Workshop Proceedings, vol. 2277, pp. 245–249. CEUR-WS.org (2018), https://ceur-ws.org/Vol-2277/paper42.pdf
10. Calamo, M., Mecella, M., Snoeck, M.: Assessing the Suitability of Large Language Models in Generating UML Class Diagrams as Conceptual Models. In: Enterprise, Business-Process and Information Systems Modeling. pp. 211–226. Springer, Cham (2025)
11. 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 MODELS. pp. 162–172. IEEE (2023)
12. Cámara, J., Burgueño, L., Troya, J.: Towards Standardized benchmarks of LLMs in software modeling tasks: a conceptual framework. Software and Systems Modeling **23**(6), 1309–1318 (Dec 2024)
13. 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 (Jun 2023)
14. De Bari, D., Garaccione, G., Coppola, R., Torchiano, M., Ardito, L.: Evaluating Large Language Models in Exercises of UML Class Diagram Modeling. In:

ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 393–399. ESEM '24, ACM, New York, NY, USA (2024)
15. Euzenat, J.: Semantic precision and recall for ontology alignment evaluation. In: Proceedings of the 20th international joint conference on Artifical intelligence. pp. 348–353. IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (Jan 2007)
16. Fettke, P., Fill, H.G., Köpke, J.: LLM, LAM, LxM Agent: From Talking to Acting Machines: Insights from the Perspective of Conceptual Modeling. Enterprise Modelling and Information Systems Architectures (EMISAJ) - International Journal of Conceptual Modeling **20** (May 2025). https://doi.org/10.18417/emisa.20.3
17. Fill, H.G.: Semantic Annotations of Enterprise Models for Supporting the Evolution of Model-Driven Organizations. Enterprise Modelling and Information Systems Architectures (EMISAJ) - International Journal of Conceptual Modeling **13**, 5:1–25 (Apr 2018)
18. Fill, H.G., Fettke, P., Köpke, J.: Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. **18**, 3 (2023). https://doi.org/10.18417/EMISA.18.3
19. Fill, H.G., Horkoff, J., Fettke, P., Köpke, J.: Generative AI and Conceptual Modeling. Business & Information Systems Engineering **68**(1), 1–5 (Feb 2026). https://doi.org/10.1007/s12599-025-00978-8, https://doi.org/10.1007/s12599-025-00978-8
20. Fill, H.G., Härer, F., Vasic, I., Borcard, D., Reitemeyer, B., Muff, F., Curty, S., Bühlmann, M.: CMAG: A Framework for Conceptual Model Augmented Generative Artificial Intelligence. In: Companion Proceedings of the 43rd International Conference on Conceptual Modeling. pp. 56–69. CEUR-WS.org (2024), https://ceur-ws.org/Vol-3849/forum5.pdf
21. Fill, H.G., Redmond, T., Karagiannis, D.: FDMM: A Formalism for Describing ADOxx Meta Models and Models. In: ICEIS 2012. pp. 133–144. SciTePress (2012)
22. Gutierrez, C., Sequeda, J.F.: Knowledge graphs. Commun. ACM **64**(3), 96–104 (2021). https://doi.org/10.1145/3418294
23. Huang, H., Chen, C., Sheng, Z., Li, Y., Zhang, W.: Can LLMs be Good Graph Judge for Knowledge Graph Construction? In: Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing. pp. 10940–10959. ACL, Suzhou, China (Nov 2025)
24. Ibáñez, M.B., Barrón-Estrada, M.L., Zatarain-Cabada, R.: Can Multimodal Large Language Models Grade Like an Expert? A Study on UML Class Diagram Assessment Accuracy. Computer Applications in Engineering Education **33**(5), e70080 (2025)
25. Johannsen, F., Fill, H.G.: Meta Modeling for Business Process Improvement. Business & Information Systems Engineering **59**(4), 251–275 (Aug 2017). https://doi.org/10.1007/s12599-017-0477-1
26. Krogstie, J.: Quality of business process models. In: The Practice of Enterprise Modeling. vol. 134, p. 76–90. Springer (2012)
27. Lankhorst, M., et al.: Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, 3 edn. (2013). https://doi.org/10.1007/978-3-642-29651-2
28. Lo, A., Jiang, A.Q., Li, W., Jamnik, M.: End-to-End Ontology Learning with Large Language Models. In: Globersons, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J.M., Zhang, C. (eds.) Advances in Neural Information Processing Systems (2024)

29. Mayr, H.C., Thalheim, B.: The triptych of conceptual modeling. Software and Systems Modeling **20**(1), 7–24 (Feb 2021)
30. McBride, B.: The Resource Description Framework (RDF) and its Vocabulary Description Language RDFS. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies, pp. 51–65. Springer, Berlin, Heidelberg (2004)
31. Michael, J., Bork, D., Wimmer, M., Mayr, H.C.: Quo Vadis modeling? Software and Systems Modeling **23**(1), 7–28 (Feb 2024)
32. Mylopoulos, J.: Conceptual modelling and Telos. Conceptual modelling, databases, and CASE: An integrated view of information system development pp. 49–68 (1992), publisher: Chichester, UK: John Wiley & Sons
33. The Open Group: ArchiMate® Specification, Version 3.2 (2021), https://publications.opengroup.org/c20c, document Number C20C
34. Pittl, B., Fill, H.G.: A visual modeling approach for the Semantic Web Rule Language. Semantic Web **11**(2), 361–389 (Feb 2020). https://doi.org/10.3233/SW-180340
35. Prokop, D., Stenchlák, , Škoda, P., Klímek, J., Nečaský, M.: Enhancing Domain Modeling with Pre-trained Large Language Models: An Automated Assistant for Domain Modelers. In: Conceptual Modeling. pp. 235–253. Springer, Cham (2025)
36. Pérez-Soler, S., Guerra, E., de Lara, J.: Flexible Modelling using Conversational Agents. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). pp. 478–482 (Sep 2019)
37. Reinhartz-Berger, I., Ali, S.J., Bork, D.: Leveraging LLMs for Domain Modeling: The Impact of Granularity and Strategy on Quality. In: Advanced Information Systems Engineering. pp. 3–19. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-94569-4_1
38. Sanderson, R.: Implementing Linked Art in a Multi-Modal Database for Cross-Collection Discovery. Open Library of Humanities **10**(2) (Jul 2024). https://doi.org/10.16995/olh.15407
39. Schoknecht, A., Vetter, A., Fill, H.G., Oberweis, A.: Using the Horus Method for Succeeding in Business Process Engineering Projects. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling: Concepts, Methods and Tools, pp. 127–147. Springer (2016)
40. Shehata, M., Lepore, B., Cummings, H., Parra, E.: Creating UML Class Diagrams with General-Purpose LLMs. In: IEEE VISSOFT. pp. 157–158 (Oct 2024)
41. Silva, J., Ma, Q., Cabot, J., Kelsen, P., Proper, H.A.: Application of the Tree-of-Thoughts Framework to LLM-Enabled Domain Modeling. In: Conceptual Modeling. pp. 94–111. Springer Nature Switzerland, Cham (2025)
42. Storey, V.C., Pastor, O., Guizzardi, G., Liddle, S.W., Maaß, W., Parsons, J., Ralyté, J., Santos, M.Y.: Large language models for conceptual modeling: Assessment and application potential. Data & Knowledge Engineering **160**, 102480 (Nov 2025)
43. Tsaneva, S., Dessì, D., Osborne, F., Sabou, M.: Knowledge graph validation by integrating LLMs and human-in-the-loop. Information Processing & Management **62**(5), 104145 (Sep 2025)
44. Vasic, I., Fill, H.G., Quattrini, R., Pierdicca, R.: Knowledge graphs vs. large language models: Competitors or partners in supporting virtual museums. ACM J. Comput. Cult. Herit. **18**(4) (Dec 2025). https://doi.org/10.1145/3756016
45. Zheng, L., Chiang, W.L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E.P., Zhang, H., Gonzalez, J.E., Stoica, I.: Judging LLM-as-a-judge with MT-bench and Chatbot Arena. In: Neurips. pp. 46595–46623. NIPS '23, Curran Associates Inc., Red Hook, NY, USA (Dec 2023)