A Framework for LLM-Based Conceptual Modeling: Application to BPMN Collaboration Diagrams

Aya Safan[1], Julius Köpke[1,*]

[1]University of Klagenfurt, Department of Informatics Systems, Universitätsstraße 65-67, 9020 Klagenfurt am Wörthersee, Austria
https://www.aau.at/en/isys/ics

Abstract

Generative AI has shown promising capabilities in translating textual descriptions into conceptual models, opening new possibilities for supporting the modeling process. In the domain of Business Process Modeling, several studies and prototypes have demonstrated the general applicability of Large Language Models (LLMs) for generating process models from textual descriptions and user feedback. This paper presents a framework for the conversational modeling of conceptual models. Our framework integrates correctness by design, generative AI and symbolic AI in the form of classical model checking to assist the user in the modeling process. We instantiate this framework in the domain of Business Process Modeling. This instantiation is the first approach going beyond the control flow perspective, enabling the modeling of multi-party collaborations, including data flow. In our evaluation, the tool demonstrated 100% syntactic correctness and notion compliance of the generated models. A BPMN-XML baseline achieved this in less than 30% of the cases. In an additional expert user study, the tool received strong user support; however, the usefulness of the model-checking functionality requires further investigation.

Keywords

Large Language Models, LLMs, Process Modeling, BPMN, Model Checking

1. Introduction

Generative AI, particularly in the form of LLMs, has shown promising capabilities for translating textual descriptions into conceptual models [1]. These capabilities allow new modeling techniques, such as LLM-based conversational modeling. In this approach, a user first sends an initial description to a modeling system, which then generates and displays a corresponding conceptual model. In an interactive feedback loop, the user can provide feedback to correct potential issues and to extend or refine the model. We argue that this approach has the potential to reduce entry barriers and advance the productivity of the conceptual modeling process.

In the most generic form, if an LLM-based modeling system could correctly interpret metamodeling languages such as MOF [2], it would be possible to use LLMs to generate arbitrary conceptual models from user input. However, as demonstrated in [3], current LLMs are not yet capable of performing this task reliably. While such a generic approach remains infeasible, several successful applications have emerged for specific output languages, such as ontologies [4] and Business Process Models [5, 6, 7, 8, 9]. Nevertheless, existing LLM-based conversational business process modeling prototypes are limited in scope. They typically focus on intra-organizational processes and address only the control flow perspective. Moreover, the generated models often depend on specially designed output formats that align with the current capabilities of LLMs.

While LLMs demonstrate notable strengths in processing natural language inputs, they are prone to producing modeling errors. Therefore, combining generative AI with symbolic AI represents a promising approach for developing real-world systems [10]. Models generated by LLMs often exhibit two main types of deficits: they may misrepresent the user's intended semantics, and they may include

ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France

*Corresponding author.

aya.safan@aau.at (A. Safan); julius.koepke@aau.at (J. Köpke)

0000-0002-6678-5731 (J. Köpke)

domain-independent modeling errors. While only the user can judge whether the model accurately reflects their intention, a system should guide the user in avoiding domain-independent errors. Examples for such errors include unsatisfiable classes in ontology modeling or violations of safeness and soundness properties in behavioral models [11]. To address these challenges, our framework combines LLM-based conversational modeling with model checking, enabling modelers to focus on correctly representing their intended semantics.

In this paper, we provide the following contributions: We introduce a generic framework for LLM-based conversational conceptual modeling in Section 2. We instantiate this framework for BPMN Collaboration Diagrams in Section 3 and present an implementation as a modeling tool in Section 4. In Section 5, we first evaluate the syntactic correctness and notation compliance of the initially generated models. In a second experiment, we assess the user acceptance of the tool in an expert user study. Related work is presented in Section 6 and Section 7 concludes the paper.

2. Framework for conversational conceptual modeling

2.1. Usage scenario and requirements

In our target scenario, a user provides a textual description to initiate process modeling. This system leverages an LLM to generate a visual model, which can be reviewed, corrected, and refined through conversational interaction. The usage scenario imposes strict requirements:

- **R1: Syntactic correctness and notation compliance:** The generated models must be syntactically correct and conform to the standard specification of the target notation, ensuring they can be rendered directly and provide a useful starting point for the conversation.
- **R2: Incremental model refinement through feedback:** The models returned in the feedback loop should incorporate the user feedback adequately to incrementally improve model quality.
- **R3: Deterministic rendering:** The positioning of model elements should be deterministic to reduce layout changes in the feedback loop.
- **R4: Usage of standards**: Standard modeling languages and serialization formats should be supported to allow reuse and interoperability with other tools for downstream tasks.
- **R5: Library of model checkers**: The application should allow the integration of model checkers for various purposes.

2.2. Framework

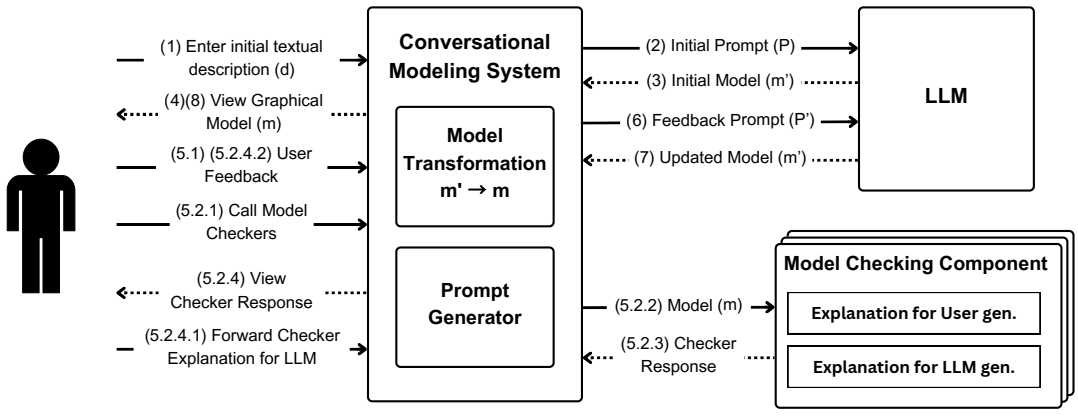

Figure 1: LLM-based conversational modeling framework with user interactions

Based on the previously identified requirements and recent studies on conversational LLM-based process modeling [5, 6, 9], we now define a generic framework for LLM-based conversational modeling,

as illustrated in Figure 1. We present the framework in the context of the usage scenario: A user sends an initial textual description d to the system to obtain a model instance of a metamodel M. The system then generates a prompt $P = (I_C, I_{F_{M'}}, , d)$, where I_C is a context-setting instruction prompt, and $I_{F_{M'}}$ instructs the LLM to produce an output as an instance of M' in the serialization format $F_{M'}$. The metamodel M' must permit an equivalent or subset of the instantiations of M. In the next step, the system transforms the model produced by the LLM to an instance of M, which includes the deterministic positioning of model elements. The generated model is displayed to the user using standard visualization components.

The user can then either send their feedback or invite model checkers to join the conversation. These are provided by a library of model checking components, each capable of accepting models in F_M or $F_{M'}$ as input. A model checking component consists of a model checker and a module for generating explanations for detected issues. For each identified issue, the component returns a set of tuples of the form (t, E), where t denotes the issue type (error, warning), and E is a set of tuples (ME, mu, ml). Here, ME contains a set of responsible model elements, mu is an explanation for the user, and ml is a more elaborate explanation for the LLM that may include technical details and suggested solutions. This structure allows for providing multiple explanations for a given issue. The system then presents the message type t and the messages mu in the chat and allows the system to highlight the elements in ME within the model.

The user can then write a custom feedback or forward ml to the LLM. We refer to the latter option as an auto-fix. Once the user sends a response, a feedback prompt $P' = (I', I_{F_{M'}}, c, S)$ is sent to the LLM. I' is an instruction for adopting the given model, S is the context, including the previously generated model as an instance of $F_{M'}$, and c is a user or auto-fix feedback. Once the user is satisfied with the generated model, it can be exported in a standard serialization format.

This framework addresses the requirements as follows: M' and its serialization $F_{M'}$ are specifically tailored to obtain syntactically correct output models (R1). To support R2, user feedback is used to iteratively refine the model across multiple interactions, enabling incremental improvement in model quality. The rendering of the graphical diagrams is realized via a deterministic algorithm and not by the LLM (R3). R4 and R5 are covered by using standard output formats and a standardized interface for model checking components.

3. Instantiation for BPMN collaboration diagrams

We now instantiate the proposed framework for the conversational modeling of BPMN Collaboration Diagrams. These diagrams capture the internal control and data flow in each pool, as well as communications between pools via message flows. We choose BPMN collaboration diagrams because they introduce an additional layer of complexity compared to models that focus only on the control flow perspective. This makes them a more realistic and demanding use case for evaluating our approach. Accordingly, the metamodel M in our framework corresponds to the BPMN 2.0 Collaboration specification [12]. The serialization format used for both rendering components and model checkers is BPMN-XML, enabling the reuse of existing tools for visualization (e.g., Camunda Modeler) and model checkers. In the following, we describe how M' and F'_M are instantiated, and discuss the role of model checking in this setting.

3.1. Intermediate process metamodel M'

While the proposed framework generally allows the same metamodel for both M and M', related work [8, 5] suggests that constraining LLM output to block-structured processes [13] significantly improves the syntactic and semantic correctness of the generated models. The restriction to block-structured processes results in a slight reduction of expressivity. However, it also avoids substantial problems arising from unstructured process models with data [14]. Additionally, block-structured processes offer correctness by design for the control-flow perspective, ensuring that process instances are safe and sound [15]. Overall, we argue that block-structured processes provide an excellent balance between

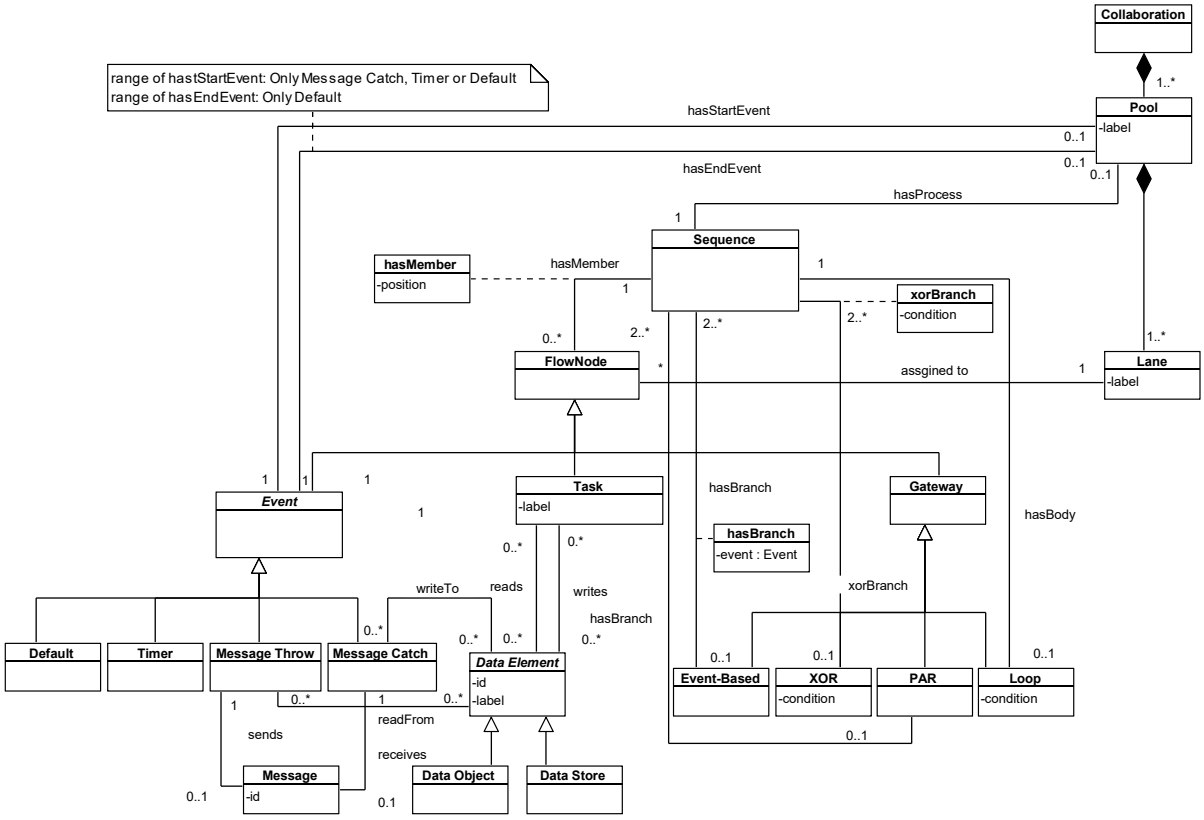

Figure 2: Metamodel of the intermediate process model M'

expressivity and correctness. Accordingly, we base our metamodel on block-structured process models. We extend the metamodel of our previous work in [5], which supports only control flow aspects of intra-organizational processes, including XOR, PAR, and LOOP blocks. Our extended metamodel is shown in Figure 2. To support BPMN Collaborations, we introduce multiple pools, each containing a block-structured process, with each flow node assigned to exactly one lane. Communication between pools is realized with messages; Message flows are modeled by referencing the identifier of the catch event from the corresponding throw event. To support passive decisions, event-based gateways are included in the model. Data handling is represented via data objects and data stores, with data flows expressed through read and write associations connected to tasks or events. Specifically, tasks can read from and write to data elements; throw events can read data for sending it to other pools, while catch events can write to local data elements. Additionally, timer events are supported.

3.1.1. Intermediate serialization format $F_{M'}$

Current LLMs, such as GPT-4.1, are optimized for agent-based interactions. A key enabler is the support for tool calling [16], where available tools and their formal input and output parameters are sent alongside the prompt. Such LLMs can reliably generate output that conforms to a specified JSON Schema. Accordingly, we defined a serialization format $F_{M'}$ of M' in the form of an annotated JSON Schema. The metamodel is nested with sequences on every level. This allows a simplified JSON representation for LLMs, where sequences are represented as arrays. The JSON Schema can be found in our evaluation repository (see Sect. 5).

3.2. Model checking components

Following the metamodel M', the control flow within a single pool is safe and sound by design [11]. Safeness ensures that each place in the Petri-net representation holds at most one token, preventing

token multiplication due to improper synchronization. Soundness requires that the process can always reach its end state without any dead transitions. Block-structured process models inherently guarantee both properties. When extended to collaborations where each process is block-structured, as required by our metamodel, safeness remains guaranteed by design [17]. However, this does not hold for soundness. To address this, we include a checker that verifies safeness and soundness of multi-pool collaborations. Data flow also introduces potential issues [18, 19], such as missing data, redundancies, or inconsistencies due to race conditions. To detect such anomalies, we include a data flow checker in our instantiation.

4. Implementation

We have implemented our instantiation of the framework as a web-based tool[1] based on the React framework. The following subsections describe the implementation of its core components.

4.1. Prompt generation

This component is responsible for translating user input into LLM API calls. While our earlier approach in [5] successfully employed zero-shot prompting for control flow modeling, this strategy proved insufficient for modeling BPMN collaborations. We have therefore opted for few-shot prompting, which has been successfully used for generating complex syntactic structures [21] and consistent models [22]. The context-setting instruction prompt contains a brief role description and explains the correct usage of selected modeling elements such as data objects, data stores, event-based gateways, and events, and provides examples in the JSON format. These examples are deliberately minimal and designed not to model full processes but to demonstrate the correct usage of specific BPMN elements as defined by our output format. The prompt is shown in Listing 1. During the feedback loop, this prompt may be overridden by a prompt provided by a model checking component. Each prompt is accompanied by an annotated JSON Schema that defines the reply format (see Subsection 3.1.1).

> You are a business process modeling expert. Use the provided textual description of a business process to generate or edit JSON process models.
>
> Pools represent separate organizations. Lanes are departments or participants within a pool. Use send and receive message events to communicate between pools only and not lanes.
> *Example*: Customer sends a request to a company.
> *Model*: ...
>
> Use send and receive message events as one-to-one. Use exclusive gateways to choose one path based on a condition. Use event-based gateways to choose a path based on incoming events.
> *Example*: Company prepares and sends an offer to Customer, who decides to accept or reject it. If no response is received within 10 days, the Company initiates a follow-up.
> *Model*: ...
>
> Use data objects, not message events, to communicate between lanes of the same pool.
> *Example*: The sales department sends a report to finance.
> *Model*: ...
>
> Use data files for digital or physical files and data stores for persistent data stores and database systems.
> *Example*: The retailer receives an order and checks the inventory database.
> *Model*: ...

Listing 1: Few-shot instructions prompt.

4.2. Model transformation

This component consists of an implemented algorithm that transforms the intermediate process model into a BPMN-XML representation for rendering, exporting, and model checking. It also deterministically

[1] https://isys.uni-klu.ac.at/pubserv/BPMN-Chatbot/v2/
The modeling tool was presented at the BPM 2025 demo session [20]. However, [20] provides only a high-level overview of the tool's modeling features and does not include architectural details, the general framework, prompts, meta-model, or evaluation.

assigns graphical coordinates to model elements, satisfying R3.

We focus here on coordinate generation. Each pool is processed independently, exploiting the block-structured form of the control flow in each pool. Relative coordinates for each flow node are established first, starting from fixed initial positions and recursively advancing x and y values. In a final pass, each node's vertical position is offset by the absolute y starting position of its respective lane. Finally, sequence and message flows are positioned based on the coordinates of their connected elements.

To support data objects, data associations are included in the model. However, in the current implementation their corresponding visual BPMN edges are intentionally omitted to avoid visual clutter and overlapping lines. Instead, annotations summarize which data objects are read or written. All data objects are rendered at the end of the pool for a clean and consistent presentation. Figure 3 shows an example process rendered by the tool.

4.3. Model checking components

Following our framework, model checkers are encapsulated as model checking components, each comprising a checker and an explanation generator. To support an extensible library of such components, they are implemented as independent REST services. Service discovery is managed through a central registry: components register themselves with metadata on startup and deregister on shutdown. The modeling system queries the registry to retrieve available components and enables users to configure which ones to use. This architecture fulfills R5 and allows for future extensions of additional checkers.

The current implementation includes two model checking components: one for assessing the safeness and soundness of BPMN Collaborations, based on the S³ checker [23], and another for validating data flow, leveraging the viadee Process Application Validator (vPAV) [24]. Each component is implemented as a Spring Boot application that invokes the underlying model checker. For each detected error, the checker returns a context-setting instruction prompt that defines the LLM's role as an expert on the specific error type and includes relevant definitions, along with multiple diagnostic explanations. Each diagnostic explanation references the relevant model elements and provides two separate descriptions: one tailored for the user and another for the LLM. When the user opts for an "auto-fix," the instruction prompt and the LLM-specific description from the diagnostic explanation are sent to the LLM, forming the feedback prompt P' (see Sect. 2.2).

Diagnostic explanations are generated by one of two modules. The generic explanation module generates predefined descriptions based on static explanatory texts associated with each error type, including potential fixes. This approach relies entirely on the LLM's reasoning capabilities to resolve the issue based on the given input. In contrast, the pattern-based explanation module offers more targeted instructions by recognizing specific (anti-)patterns linked to known issues. For instance, an unsafe process may be further explained if a message is conditionally sent in one pool but unconditionally received in another active pool. In such cases, the error message includes possible fixes based on the identified pattern.

4.3.1. User interaction

When a user invites model checkers to the chat, they are called in parallel. All identified errors and warnings are presented as chat messages from the respective checking components. If model elements related to the issues are provided, users can highlight them directly in the graphical model. Finally, the user can either write their own feedback or use the auto-fix functionality. Figure 3 shows an example of such an interaction from the tool.

5. Evaluation

We evaluate our framework's instantiation against the requirements outlined in Subsection 2.1. Specifically, we assess the syntactic correctness and notation compliance of the generated models (R1). A key aspect of our framework is the separation between the intermediate representation used by the LLM

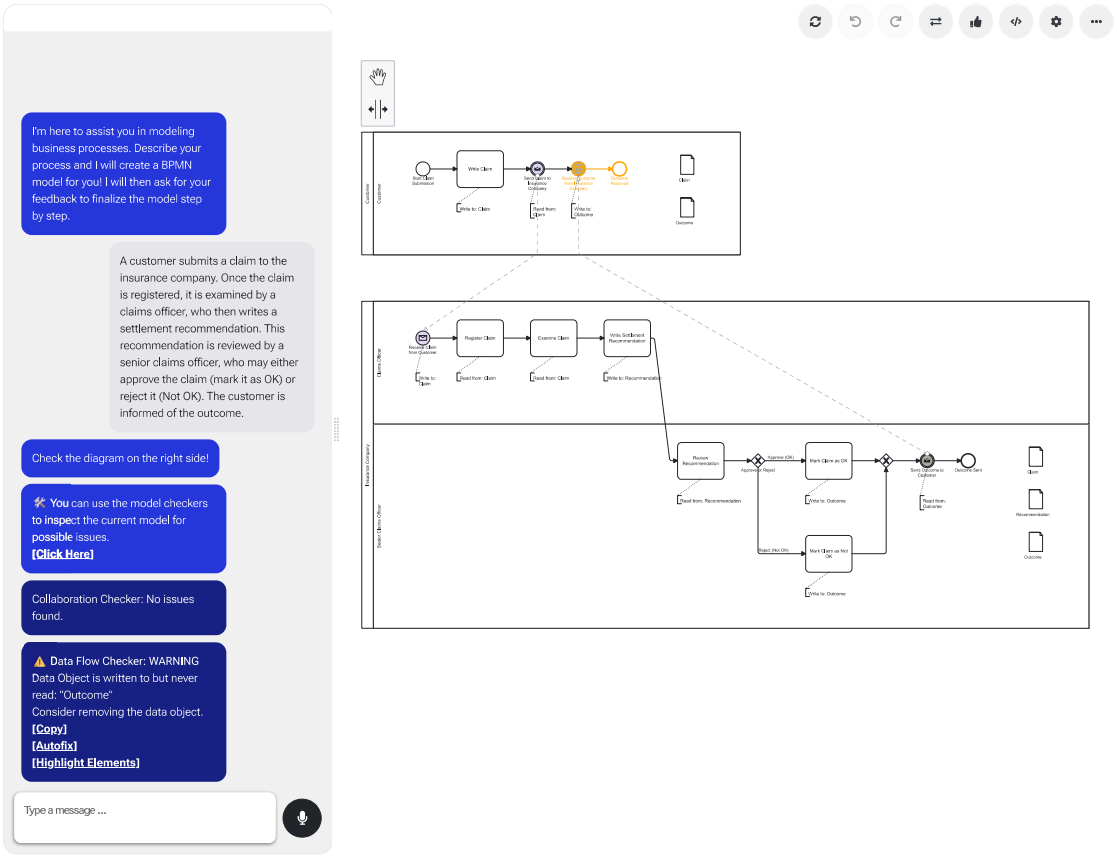

Figure 3: Tool interface with the warning from the checker highlighted.

and the final output format. To validate this design decision, we compare the syntactic correctness and notation compliance of our generated models to that of a direct BPMN-XML generation baseline. This approach is further justified by the lack of existing baselines supporting multiple pools and data flow.

Another key assumption in our framework is that interactive modeling with a feedback loop improves the modeling process and generated model quality (R2). We therefore conduct an expert user study to assess the general usefulness of the tool, and the usefulness of the model checkers and auto-fix feature in particular. To the best of our knowledge, with the exception of our work in [5], prior approaches supporting feedback loops [6, 9] have not been evaluated in this regard. All generated models, prompts, the JSON-Schema for LLM responses, and user study results are available in our repository[2].

5.1. Dataset

We compiled a dataset by collecting publicly available process descriptions from [25, 26, 27, 28, 29, 30], resulting in 59 descriptions in total. We excluded descriptions from the PET Dataset [31], as it was used during our initial prompt tuning and early-stage testing, and due to its widespread use, it imposes the risk of data leakage. Additionally, we filtered process descriptions that required BPMN elements not currently supported by our schema, such as boundary events. Descriptions exceeding 500 words, or roughly one page, were excluded as their length would impose an unreasonable burden on expert reviewers. After applying these criteria, 49 descriptions remained.

Since neither LLMs nor experts produce deterministic outputs, we generated three models for each description, and each such model was evaluated by 2 experts. Including the baseline, this leads to $n * 2 * 2 * 3$ evaluations. We therefore restrict n to 6, and we evaluate with one LLM, leading to 72 model evaluations. To ensure diversity, we clustered the descriptions into three groups based on complexity

[2] https://anonymous.4open.science/r/ER2025-5134/

and modeling requirements. This clustering considered metrics such as the words and sentences count, the estimated number of BPMN elements needed (e.g., pools, lanes, data objects, exclusive, parallel, loop blocks), and the nesting depth. Two descriptions were then randomly selected from each cluster, as listed in Table 1.

Case	Example	#W	#S	#P	#L	#D	#Dec	#Par	#Loop	Depth	Cluster
1	0 [28]	66	4	2	2	2	1	1	0	2	0
2	4 [29]	50	2	1	2	1	1	0	0	1	0
3	V_k09.txt [30]	161	9	1	2	5	2	1	1	3	1
4	4 [28]	176	8	2	2	4	1	0	0	1	1
5	R_j04.txt [30]	124	10	3	3	8	2	1	1	2	2
6	11 [26]	312	21	3	5	4	2	1	0	2	2

Table 1: Selected process descriptions and their features. Abbreviations: W = words, S = sentences, P = pools, L = lanes, D = data objects, Dec = decisions, Par = parallel constructs, Loop = loops, Depth = nesting depth.

5.2. Initial model generation

To establish a baseline, we use a prompt designed to instruct the LLM to produce BPMN-XML directly. This prompt was adapted from our original instruction prompt. It includes the same available modeling elements, usage explanations, and examples. We further instructed the LLM to exclude the BPMN diagram visualization from the output, as it falls outside the evaluation scope and is generated separately.

We generated models for both our tool and the baseline for each of the 6 textual process descriptions. Each description was processed in three separate runs on different days to avoid prompt caching[3]. The experiments were executed between April 23 and April 25 using OpenAI GPT-4.1 with a temperature of 0.2, chosen to reduce randomness and ensure more consistent and reproducible outputs. The same settings were also used for the second experiment. This has led to 18 models produced by our tool and 18 models produced by the BPMN-XML baseline.

5.3. Experiment 1: Syntactic Correctness and Notation Compliance

This experiment evaluates the syntactic correctness and compliance of initial process models generated by our approach, compared to the baseline. We first assessed syntactic correctness, ensuring all models are well-formed XML and fully conform to the BPMN 2.0 XML Schema, with no duplicate or invalid tags or attributes. We then manually assessed correct use of the modeling language, verifying that each model adheres to the structural and behavioral constraints defined by the BPMN 2.0 specification [12].

The evaluation results are summarized in Table 2. Our approach achieved 100% syntactic correctness and correct use of the modeling language, compared to only 22%, respectively 28% for the baseline. Common baseline issues included start events without outgoing sequence flows, end events without incoming flows, use of message flows within a single pool, message flows connected to generic start events, and data objects incorrectly used as targets of sequence flows. The results indicate that LLMs struggle to maintain BPMN's strict syntactic and semantic rules when generating XML directly. This highlights the benefits of using an intermediate output format and metamodel as proposed in our framework. Unlike direct XML, our approach clearly meets requirement R1 for conversational modeling.

Metric	Our Approach	Baseline
Syntactic Correctness	100% (18 of 18)	22% (4 of 18)
Correct Use of Modeling Language	100% (18 of 18)	28% (5 of 18)

Table 2: Results for initially generated models

[3] https://platform.openai.com/docs/guides/prompt-caching

5.4. Experiment 2: Expert user study

We base our experiments on an expert user study with eighteen experts in process modeling. In this experiment, we evaluate the tool based on a technology acceptance test [32], focusing on perceived usefulness, ease of use, and intention to use. We also assess whether the integration of model checkers, including the auto-fix feature, is a useful feature for modelers and whether the tool provides valuable responses to feedback. This is measured by survey questions using a 7-point Likert scale and by an examination of the gain in perceived model quality after the usage of the tool.

Therefore, each expert first reviewed two models generated by our tool and assessed the perceived semantic quality (PSQ) [33] based on the refined framework for process models [34]. We slightly adapted the questions to our setting; The term "conceptual model" was replaced by "generated model". The indicators used in this evaluation, along with their definitions and associated statements, are presented in Table 3. We excluded the perceived authenticity measure, as it reflects the quality of the textual description rather than that of the model.

After the initial assessment, each expert used the tool to continue working on one of the previously assessed models, with explicit instructions to also use the auto-fix feature. After completing the modeling task, they again rated the perceived quality of their resulting model on the same 7-point Likert scale and answered questions about the usefulness of the model checkers, the auto-fix feature, and the tool's responses to feedback, as well as the degree of frustration during the experiment. Finally, the experts completed the technology acceptance questions.

Indicator	Definition	Statement
Correctness	All statements in the representation are correct.	*"The generated model represents the business process correctly."*
Relevance	All statements in the representation are relevant to the problem.	*"All the elements in the generated model are relevant for the representation of the business process."*
Completeness	The representation contains all statements about the domain that are correct and relevant.	*"The generated model gives a complete representation of the business process."*

Table 3: PSQ indicators, definitions, and statements (adapted from [34])

Demographics: Out of the initial 18 candidates, 17 successfully finished the modeling tasks. Six participants were professors, 10 were post-docs, and 2 participants were PhD candidates. Participants included 13 males and five females from 11 universities; all had a strong background in process modeling, either as part of the BPM community or by teaching BPMN-related courses.

5.4.1. Results

Technology Acceptance Regarding the acceptance of the tool, the experts gave a median rating of 6 for overall perceived usefulness, and 5 for intention to use, ease of use, and perceived productivity. Details on the distribution are shown in Figure 4. For ease of use, 15 out of 17 participants rated the tool 5 or higher, and 12 perceived an increase in productivity. Overall usefulness received particularly positive feedback, with 13 responses assigning ratings of 6 or 7. Although the intention to use showed a broader distribution, 12 participants still rated it 5 or above, indicating a generally positive inclination toward future adoption.

Indirect Evaluation of the Feedback Loop The expert ratings of the initial models produced median scores of 5 for correctness, 6 for relevance, and 6 for completeness. We compare the quality assessments provided by the experts for the initial models with those made after they finished the modeling task using the tool. The analysis is based on the normalized change score, which standardizes relative improvement by considering both the initial evaluation scores and the maximum possible gain.

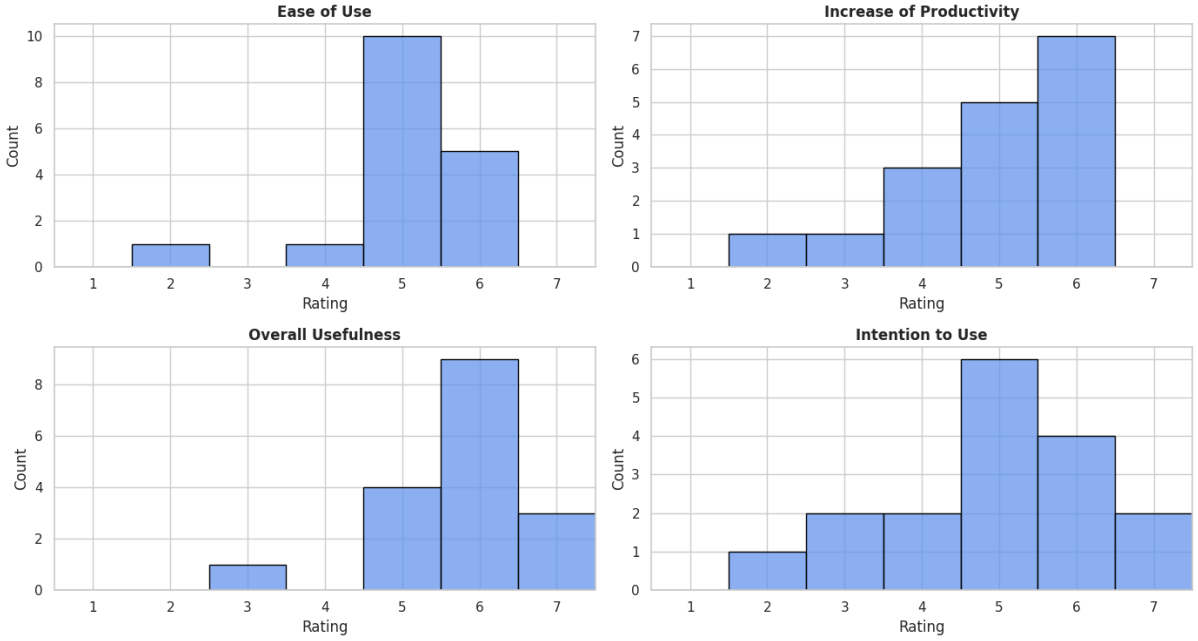

Figure 4: Distribution of Participant Ratings on Technology Acceptance

On average, we observed consistent improvements across all quality dimensions: correctness increased by **36%**, relevance by **39%**, and completeness by **28%**. These results suggest that expert involvement, aided by the tool, generally led to enhanced model quality, thereby fulfilling requirement R2.

Usefulness of model checkers and Auto-Fix Experts rated the usefulness of the model checkers at a neutral median (4), while the auto-fix feature received a lower median score of 3. The tool feedback resulted in a median of 4, whereas frustration levels were notably low, with a median of 2. Since the experts were asked to assess the technology acceptance questions based on their experiences when using the tool, the high technology acceptance scores and low frustration suggest a generally positive user experience. However, the relatively lower rating for the auto-fix feature may have contributed to the low direct assessment of the feedback of the tool. One possible explanation is that experts were explicitly informed before using the tool that they would be asked about the model checkers and auto-fix functionality. This may have led to an overemphasis on these features. Moreover, some experts tried to achieve impossible modeling goals, such as changing the process model to a non-block-structured one, changing the representation of data associations, collapsing pools, or asking for textual responses. All those features are not currently supported by the tool.

Overall, the high ratings for technology acceptance, low frustration levels, and the observed gains in model quality support the usefulness of the system's responses. However, there is no clear indication of the usefulness of model checkers and the auto-fix feature. Further investigation into the success rates of auto-fixes is needed, and potentially, a more sophisticated diagnosis and repair generation is required. The underlying hypothesis of our implementation, that the LLM can repair a model based on a diagnosis and a definition, only partially held in our experiments. Nevertheless, this limitation is specific to our current implementation and not the framework in general, and requires further investigation.

Threats to validity The expert user study approach imposes limitations on the feasible sample size, as volunteering experts are typically unwilling to evaluate a large number of models. Also, the non-deterministic nature of LLM outputs necessitates evaluating multiple solutions per process. Our restriction to 18 modeling experts and 6 process models limits the generalizability of our findings to other processes and user groups.

Furthermore, relying on textual descriptions as a proxy for the modeler's domain knowledge introduces limitations, particularly in light of the SEQUAL framework's emphasis on interpretation and knowledge alignment [35, 36]. Additionally, in interactive modeling scenarios, the understanding of the domain may evolve during the modeling process. This highlights the need for further studies on this dynamic setting. In particular, it would be valuable to investigate how the implicit knowledge embedded in the LLM can support users during modeling.

6. Related Works

LLMs are increasingly being explored for their potential to support conceptual modeling in information systems development [37]. Beyond their well-known capabilities in text and code generation, recent studies show that LLMs can assist in creating conceptual models from natural language input. They have been used to extract domain entities and relationships [38], facilitate ontology extraction [39], and generate suggestions for designing classes, attributes, and associations [40]. Additionally, LLMs have shown promise in extracting process elements, such as activities, actors, and relations, from textual descriptions [22]. Overall, LLMs are emerging as valuable assistants for conceptual modeling.

Even before the rise of LLMs, classical NLP techniques were applied to derive process models from textual descriptions. For example, [29] used traditional NLP methods to extract fact types from text and map them into spreadsheet-based BPMN representations, including data flow. Recent efforts shift toward leveraging LLMs. For example, [41] evaluates open-source LLMs for generating BPMN-XML from process descriptions, finding that smaller language models frequently fail to produce valid BPMN-XML, even when templates are provided, limiting practical applicability. To address quality and syntactic limitations, several approaches are based on intermediate representations to improve model quality. [8] proposes a fine-tuned GPT model that generates regex-like expressions representing simple control flow, which are parsed into abstract syntax trees and translated into BPMN. Similarly, [7] extracts activities and their associated dependencies and conditions as structured JSON from text to automate BPMN generation, including lane assignment.

Other works present actual tools that incorporate iterative refinement. For instance, ProMoAI [6] uses LLMs with extensive prompts to produce Python code representing partially ordered workflow nets (POWL), which are rendered as BPMN or PNML. Nala2BPMN [9] leverages the LangChain framework to use LLMs for key entities, dependency and branching relations extraction to construct BPMN models. Our previous tool [5] converts user input into BPMN models via an intermediate JSON format.

All these previous LLM-based approaches focus on generating the control flow perspective. In this paper, we have presented in detail the first approach for conversational process modeling, going beyond these limitations, supporting multi-pool BPMN collaboration diagrams. Surprisingly, to the best of our knowledge, with the exception of our own previous work, all other works supporting a feedback loop have only provided evaluations of initial models.

7. Conclusion and future works

In this paper, we introduced a generic framework for LLM-based conversational conceptual modeling. It allows for distinguishing between the metamodel and serialization format used for the LLM and those used for user interactions and other components. This distinction allows tuning for the capabilities of the used LLM. Furthermore, it integrates classical model checkers into the conversational modeling process, allowing them to take part in the chat and report on errors. Importantly, the representation of these errors provided to the user and to the LLM can differ, enabling tailored responses optimized for LLM usage.

We have instantiated this framework for the generation of BPMN collaboration diagrams. The developed tool is the first LLM-based approach in the domain of process modeling that goes beyond the control flow by supporting data objects, data flows, and multiple pools. The evaluations of the initial models clearly demonstrate the benefit of using a custom metamodel and serialization format.

Our implementation produces process models that fully adhere to the correct syntax and language usage. The XML baseline achieved this in only 22% and 28% of the cases. An overall assessment of the acceptance of the tool indicates that the tool not only meets the requirements for conversational modeling but also achieves strong user support. This tool can be utilized in various scenarios, as a stand-alone system, be combined with traditional modeling systems for manual refinements, or receive its input from an upstream LLM-based pipeline.

However, the implemented model checking components did not achieve comparable user support. Future works, therefore, include a detailed investigation of the auto-fix capabilities. It is of particular interest how classical model repair methods [42] can be combined with LLMs. Nevertheless, this current limitation of the tool does not apply to the framework in general. To assess scalability in complex, real-world scenarios, we also plan an evaluation with industry users. Additionally, we aim to instantiate the framework for other modeling languages, such as UML class diagrams with OCL constraints and ontologies. Other future works include combining the framework with LLM agents [43].

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly to check grammar and spelling, paraphrase, and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] H.-G. Fill, P. Fettke, J. Köpke, Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT, Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 18 (2023) 3. doi:10.18417/EMISA.18.3.

[2] Object Management Group, Meta Object Facility (MOF) Specification, ISO/IEC 19502:2005, https://www.omg.org/spec/MOF/ISO/19502/PDF, 2005. Accessed: 2025-05-23.

[3] F. Muff, H.-G. Fill, Limitations of chatgpt in conceptual modeling: Insights from experiments in metamodeling, in: Modellierung 2024 Satellite Events, Gesellschaft für Informatik e.V., 2024. doi:10.18420/modellierung2024-ws-008.

[4] M. P. Joachimiak, M. A. Miller, J. H. Caufield, R. Ly, N. L. Harris, A. Tritt, C. J. Mungall, K. E. Bouchard, The artificial intelligence ontology: Llm-assisted construction of ai concept hierarchies, Applied Ontology 19 (2024) 408–418. doi:10.1177/15705838241304103. arXiv:https://doi.org/10.1177/15705838241304103.

[5] J. Köpke, A. Safan, Efficient llm-based conversational process modeling, in: K. Gdowska, M. T. Gómez-López, J. Rehse (Eds.), Business Process Management Workshops - BPM 2024 International Workshops, Krakow, Poland, September 1-6, 2024, Revised Selected Papers, volume 534 of *Lecture Notes in Business Information Processing*, Springer, 2024, pp. 259–270. doi:10.1007/978-3-031-78666-2_20.

[6] H. Kourani, A. Berti, D. Schuster, W. M. P. van der Aalst, Process modeling with large language models, in: Enterprise, Business-Process and Information Systems Modeling, Springer Nature Switzerland, Cham, 2024, pp. 229–244.

[7] F. Ajmal, P. Wijekoon, H. Dhanamina, Y. Ravishan, D. Nawinna, B. Attanayaka, Automated bpmn diagram generation, in: 2024 6th International Conference on Advancements in Computing (ICAC), 2024, pp. 7–12. doi:10.1109/ICAC64487.2024.10851120.

[8] Q. Nivon, G. Salaün, Automated generation of bpmn processes from textual requirements, in: W. Gaaloul, M. Sheng, Q. Yu, S. Yangui (Eds.), Service-Oriented Computing, Springer Nature Singapore, Singapore, 2025, pp. 185–201.

[9] A. Nour Eldin, N. Assy, O. Anesini, B. Dalmas, W. Gaaloul, Nala2bpmn: Automating bpmn model generation with large language models, in: Cooperative Information Systems: 30th International

Conference, CoopIS 2024, Porto, Portugal, November 19–21, 2024, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2025, pp. 398–404. URL: https://doi.org/10.1007/978-3-031-81375-7_27.

[10] R. Buchmann, J. Eder, H. Fill, U. Frank, D. Karagiannis, E. Laurenzi, J. Mylopoulos, D. Plexousakis, M. Y. Santos, Large language models: Expectations for semantics-driven systems engineering, Data Knowl. Eng. 152 (2024) 102324. doi:10.1016/J.DATAK.2024.102324.

[11] W. M. P. van der Aalst, Workflow Verification: Finding Control-Flow Errors Using Petri-Net-Based Techniques, Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 161–183.

[12] Object Management Group, Business process model and notation (bpmn) version 2.0, https://www.omg.org/spec/BPMN/2.0/, 2011. OMG Document Number: formal/2011-01-03.

[13] O. Kopp, D. Martin, D. Wutke, F. Leymann, The difference between graph-based and block-structured business process modelling languages, Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 4 (2009) 3–13. doi:10.18417/EMISA.4.1.1.

[14] C. Combi, M. Gambini, Flaws in the flow: The weakness of unstructured business process modeling languages dealing with data, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2009, pp. 42–59.

[15] W. M. P. van der Aalst, Structural characterizations of sound workflow nets, Technical Report 9623, Technische Universiteit Eindhoven, 1996.

[16] Z. Shen, Llm with tools: A survey, 2024. URL: https://arxiv.org/abs/2409.18807. arXiv:2409.18807.

[17] F. Corradini, A. Morichetta, C. Muzi, B. Re, F. Tiezzi, Well-structuredness, safeness and soundness: A formal classification of bpmn collaborations, Journal of Logical and Algebraic Methods in Programming 119 (2021) 100630. URL: https://www.sciencedirect.com/science/article/pii/S2352220820301152. doi:https://doi.org/10.1016/j.jlamp.2020.100630.

[18] S. Sadiq, M. Orlowska, W. Sadiq, C. Foulger, Data flow and validation in workflow modelling, in: Proceedings of the 15th Australasian database conference-Volume 27, 2004, pp. 207–214.

[19] S. Von Stackelberg, S. Putze, J. Mülle, K. Böhm, Detecting data-flow errors in bpmn 2.0, Open Journal of Information Systems (OJIS) 1 (2014) 1–19.

[20] A. Safan, J. Köpke, BPMN-Chatbot++: LLM-Based Modeling of Collaboration Diagrams with Data, in: Proceedings of the BPM 2025 Demos & Resources Forum, 2025. To appear.

[21] M. B. Chaaben, L. Burgueño, H. Sahraoui, Towards using few-shot prompt learning for automating model completion, in: Proceedings of the 45th International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER '23, IEEE Press, 2023, p. 7–12. doi:10.1109/ICSE-NIER58687.2023.00008.

[22] J. Neuberger, L. Ackermann, H. van der Aa, S. Jablonski, A universal prompting strategy for extracting process model information from natural language text using large language models, in: W. Maass, H. Han, H. Yasar, N. J. Multari (Eds.), Conceptual Modeling - 43rd International Conference, ER 2024, Pittsburgh, PA, USA, October 28-31, 2024, Proceedings, volume 15238 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 38–55. doi:10.1007/978-3-031-75872-0_3.

[23] F. Corradini, A. Morichetta, A. Polini, B. Re, L. Rossi, F. Tiezzi, Correctness checking for bpmn collaborations with sub-processes, Journal of Systems and Software 166 (2020) 110594. URL: https://www.sciencedirect.com/science/article/pii/S0164121220300716. doi:https://doi.org/10.1016/j.jss.2020.110594.

[24] K. Schneid, S. D. Bernardo, H. Kuchen, S. Thöne, Data-Flow Analysis of BPMN-based Process-Driven Applications: Detecting anomalies across model and code, ERCIS Working Paper 38, Münster, 2021. URL: https://hdl.handle.net/10419/243142.

[25] O. Holschke, Impact of granularity on adjustment behavior in adaptive reuse of business process models, in: R. Hull, J. Mendling, S. Tai (Eds.), Business Process Management, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 112–127.

[26] F. Friedrich, J. Mendling, F. Puhlmann, Text2process: Process model generation from natural language text, 2015. URL: https://github.com/FabianFriedrich/Text2Process.

[27] Camunda, BPMN for Research, https://github.com/camunda/bpmn-for-research, 2015. URL: https://github.com/camunda/bpmn-for-research, a collection of BPMN diagrams created during Camunda

training sessions, intended for academic and research purposes.
[28] B. Schäfer, H. van der Aa, H. Leopold, H. Stuckenschmidt, Sketch2bpmn: Automatic recognition of hand-drawn bpmn models, in: Advanced Information Systems Engineering, Lecture Notes in Computer Science, Springer International Publishing, 2021.
[29] S. Sholiq, R. Sarno, E. S. Astuti, Generating bpmn diagram from textual requirements, Journal of King Saud University - Computer and Information Sciences 34 (2022) 10079–10093. URL: https://www.sciencedirect.com/science/article/pii/S1319157822003585. doi:https://doi.org/10.1016/j.jksuci.2022.10.007.
[30] J. Mangler, N. Klievtsova, Textual process descriptions and corresponding bpmn models, 2023. doi:10.5281/zenodo.7783492.
[31] P. Bellan, et. Al., Process extraction from natural language text: the PET dataset and annotation guidelines, in: Proceedings of NL4AI'2022), volume 3287 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 177–191. URL: https://ceur-ws.org/Vol-3287/paper18.pdf.
[32] F. Davis, F. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quarterly 13 (1989) 319–. doi:10.2307/249008.
[33] G. Poels, A. Maes, F. Gailly, R. Paemeleire, Measuring the perceived semantic quality of information models, in: J. Akoka, S. W. Liddle, I.-Y. Song, M. Bertolotto, I. Comyn-Wattiau, W.-J. van den Heuvel, M. Kolp, J. Trujillo, C. Kop, H. C. Mayr (Eds.), Perspectives in Conceptual Modeling, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 376–385.
[34] P. Rittgen, Quality and perceived usefulness of process models, in: Proceedings of the 2010 ACM Symposium on Applied Computing, 2010, pp. 65–72.
[35] J. Krogstie, G. Sindre, H. Jørgensen, Process models representing knowledge for action: a revised quality framework, Eur. J. Inf. Syst. 15 (2006) 91–102. doi:10.1057/palgrave.ejis.3000598.
[36] J. Krogstie, Quality of business process models, volume 134, 2012, pp. 76–90. doi:10.1007/978-3-642-34549-4_6.
[37] V. Storey, O. Pastor, G. Guizzardi, S. Liddle, W. Maass, J. Parsons, J. Ralyté, M. Santos, Large language models for conceptual modeling: Assessment and application potential, 2025. doi:10.2139/ssrn.5170458.
[38] T. D. Stojanović, K. Jovanović, S. D. Lazarević, Evaluation of gpt-generated conceptual models based on verbal descriptions: accuracy and quality analysis, in: 2025 29th International Conference on Information Technology (IT), 2025, pp. 1–4. doi:10.1109/IT64745.2025.10930270.
[39] M. Abolhasani, R. Pan, Leveraging llm for automated ontology extraction and knowledge graph generation, 2024. doi:10.48550/arXiv.2412.00608.
[40] D. Prokop, Š. Stenchlák, P. Škoda, J. Klímek, M. Nečaský, Enhancing domain modeling with pre-trained large language models: An automated assistant for domain modelers, in: W. Maass, H. Han, H. Yasar, N. Multari (Eds.), Conceptual Modeling, Springer Nature Switzerland, Cham, 2025, pp. 235–253.
[41] C. Lauer, P. Pfeiffer, A. Rombach, N. Mehdiyev, Conversational business process modeling using llms: Initial results and challenges, in: EMISA 2025, Gesellschaft für Informatik e.V., Bonn, 2025, pp. P3+2–O4+5. doi:10.18420/EMISA2025_03.
[42] A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, Repairing unsatisfiable concepts in owl ontologies, in: European Semantic Web Conference, Springer, 2006, pp. 170–184.
[43] P. Fettke, J. Köpke, H. Fill, Llm, lam, lxm agent: From talking to acting machines insights from the perspective of conceptual modeling, Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. 20 (2025). URL: https://doi.org/10.18417/emisa.20.3. doi:10.18417/EMISA.20.3.