BPMN-Chatbot++: LLM-Based Modeling of Collaboration Diagrams with Data

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 and large language models (LLMs) have shown promising capabilities in generating business process models from textual descriptions and interactive user feedback. In this paper, we present an extended version of the *BPMN-Chatbot*, a conversational modeling tool that now supports BPMN Collaboration diagrams with multiple pools, lanes, message flows, and data objects. The tool combines LLM-based generation with symbolic AI in the form of classical model checking to detect and explain modeling errors. A preliminary evaluation demonstrates strong user acceptance and consistently high quality of the generated models.

Keywords

Large Language Models, LLM, Conversational Process Modeling, Model Checking

1. Introduction

Recent advances in generative AI have enabled the use of LLMs for conversational business process modeling, where users describe a process in natural language and receive a corresponding model. However, [?] shows that LLMs often fail to generate valid BPMN XML directly, even with explicit formatting guidelines and a one-shot example in the prompt. To address this limitation, LLM-based conversational tools like ProMoAI [?], Nala2BPMN [?], and our previous work [?] use intermediate formats such as Python code, extracted entities, or structured JSON to construct BPMN models more reliably. Existing approaches remain limited to only the control-flow perspective in intra-organizational processes. Moreover, despite their strengths in understanding natural language, LLMs frequently introduce modeling errors.

We present an extension of the *BPMN-Chatbot*, originally introduced in [?]. According to the available literature, this extension makes it the first LLM-based conversational modeling tool to support BPMN collaboration diagrams with multiple pools, lanes, message flows, and data objects, which are key elements for modeling inter-organizational processes. The tool combines LLM-based generation with model checking to detect domain-independent modeling issues. Users iteratively refine their models through a chatbot interface, using either manual feedback or automated fixes. This has the potential to significantly streamline and democratize BPMN modeling, making it accessible to domain experts without deep technical knowledge.

2. Extending the BPMN-Chatbot

The *BPMN-Chatbot* has been extended to support BPMN collaborations with multiple pools and lanes. The tool now also supports event-based gateways for passive decisions, timer events, and data flow through data objects and data stores. The extended tool is publicly available on our homepage[1], along

Proceedings of the Best BPM Dissertation Award, Doctoral Consortium, and Demonstrations & Resources Forum co-located with 23rd International Conference on Business Process Management (BPM 2025), Seville, Spain, August 31st to September 5th

*Corresponding author.
aya.safan@aau.at (A. Safan); julius.koepke@aau.at (J. Köpke)
0000-0002-6678-5731 (J. Köpke)

[1]https://isys.uni-klu.ac.at/pubserv/BPMN-Chatbot/v2/, OpenAI API keys for reviewing the tool will be made available to the organizers upon request.

with a video demonstration and additional resources.

2.1. Architecture

The architecture of the extended *BPMN-Chatbot* tool is shown in Figure 1. The tool is implemented as a React single-page web application. We focus here on three core components: *Prompt Generation*, *Model2Model Translation*, and *Model Checkers*.

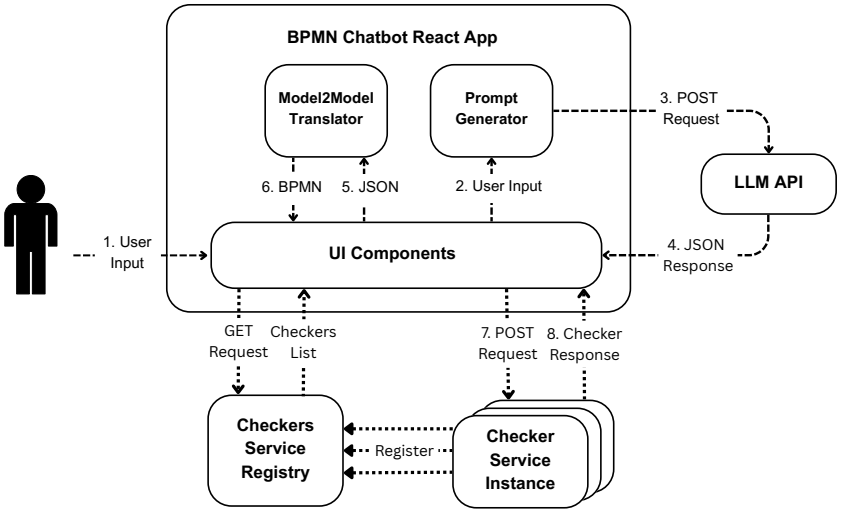

Figure 1: Architecture of the extended BPMN-Chatbot

2.1.1. Prompt Generation

This component is responsible for generating structured prompts based on user input and system context, which are then sent as API calls to the LLM. We still use an intermediate JSON format to represent the process model. The extended JSON schema is provided to the LLM through a function definition included in the tools. While our earlier approach successfully employed zero-shot prompting, it was insufficient to address the complexity of BPMN collaboration modeling, leading us to adopt few-shot prompting. Our prompt includes a brief role description and minimal examples illustrating correct usage of specific BPMN elements in our JSON format.

2.1.2. BPMN-XML Generation

The Model2Model Translator is our custom-built component that converts the intermediate block-structured process model, represented in a JSON format, into a BPMN XML representation for rendering, exporting, and model checking. It deterministically assigns graphical coordinates to flow elements based on the block-structured control flow within each pool. Nodes are positioned using relative coordinates, which are then adjusted by lane-specific vertical offsets. Sequence and message flows are added based on node positions. To reduce visual clutter, data objects and associations are included in the XML but edges are not rendered; instead, they are indicated by annotations. Figure 4 shows an example diagram rendered by the tool.

2.1.3. Model Checkers

Model checkers are implemented as independent services that analyze BPMN models and detect potential issues. Each registers itself in a central service registry at startup and deregisters at shutdown, enabling dynamic discovery. When a checker analyzes the model, it returns issues labeled as errors or warnings, along with optional metadata about relevant elements. Each issue includes two types of

explanations: one for the user and one tailored for the LLM. The LLM-specific explanation includes the issue description along with system instructions that define the LLM's expert role on the specific error type. When the user selects the "auto-fix" option, this explanation is forwarded to the LLM, enabling it to automatically resolve the issue. The current setup includes two checkers: one for assessing safeness and soundness of BPMN Collaborations (based on the S^3 checker [**?**]), and one for validating data flow (based on the viadee Process Application Validator, vPAV [**?**]). Both are implemented as Spring Boot services, wrapping the underlying checkers and generating tailored user and LLM explanations.

2.2. Usage Scenario

The tool offers a configurable settings panel where users can adjust parameters such as the underlying LLM model, temperature, instruction prompt, and the set of supported BPMN elements. Users can also browse and select from available model checkers. These options allow the tool to be tailored to specific use cases or used with default settings.

To start the modeling process, the user begins by providing a textual or voice description of a process, which is processed by an LLM to generate and visualize an initial BPMN process model. Users can then provide direct feedback or invite model checkers into the chat. These checkers run in parallel and return any identified errors or warnings as chat messages. When no issues are reported, the generated model is confirmed to be correct, as shown in Figure 2. Otherwise, users can highlight relevant model elements in the diagram and either write their own feedback or use the auto-fix functionality.

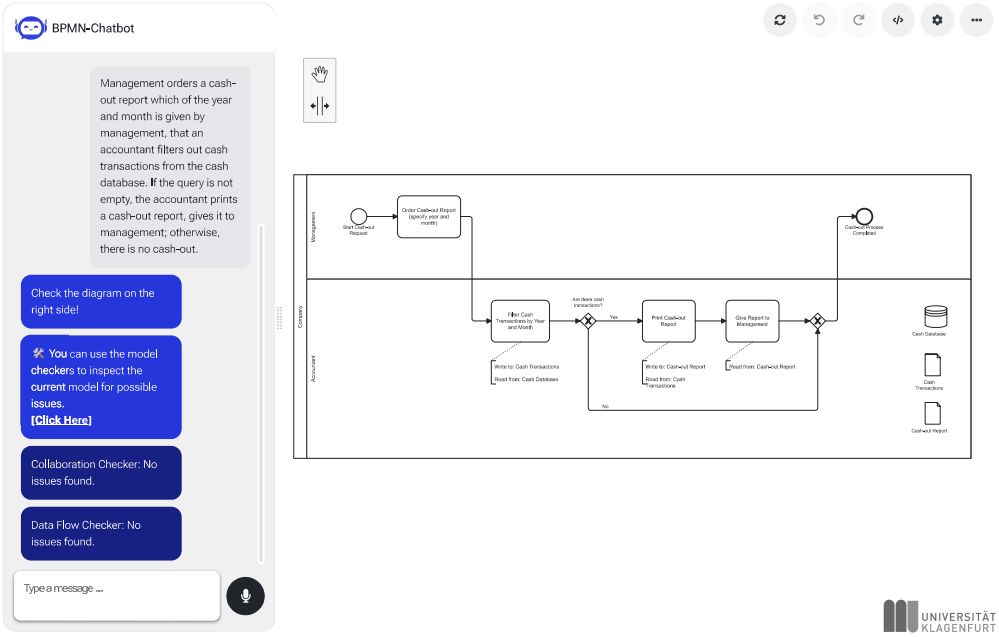

Figure 2: Initial process model with no issues detected.

Figure 3 shows an example with three messages from two model checkers. The first two messages, both generated by the collaboration checker, report a soundness error due to dead tokens. The second message elaborates on this issue by explaining that a message is conditionally sent in one pool but unconditionally received in another pool. The checker recommends restructuring the model by placing the message catch event after an event-based gateway. The third message, issued by the data flow checker, presents a warning that a data object is written to but never read, and suggests removing it.

In this scenario, the user chooses to apply the auto-fix proposed in the second message, which addresses the collaboration error. The revised model shown in Figure 4 resolves the identified issue by modifying the gateway structure as suggested. The final model can be exported as a BPMN-XML file, and the complete session state can be saved and reloaded for continued modeling at a later stage.

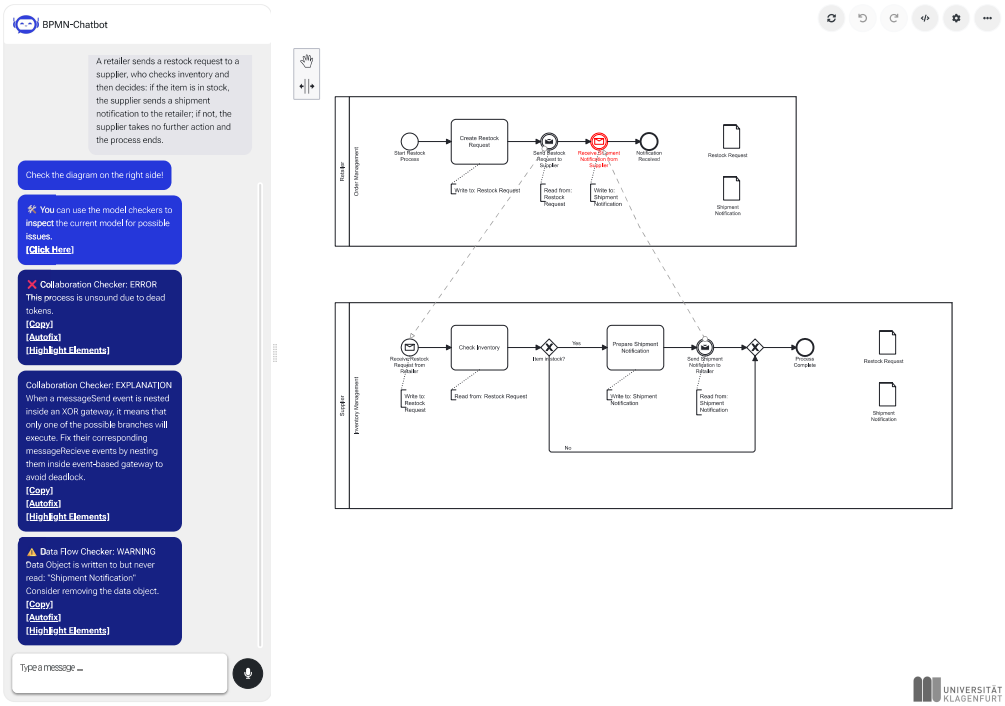

Figure 3: Tool interface with the error from the collaboration checker highlighted.

Figure 4: Updated process model after applying the auto-fix suggested by the collaboration checker.

3. Maturity of the Tool

In [**?**], the original version of the *BPMN-Chatbot* was compared against ProMoAI, alongside a technology acceptance experiment conducted with a broad audience at a science fair, demonstrating high quality of the generated models and overall tool usefulness.

For the extended tool, an evaluation is in progress. Preliminary data already indicate strong user acceptance and superior syntactic correctness and model quality compared to a baseline prompt producing XML directly. To establish the baseline, we adapted our original prompt, which generates models in our intermediate JSON format, to instead request direct BPMN XML from the LLM.

4. Conclusions and Future Works

To the best of our knowledge, this extension of the *BPMN-Chatbot* presents the first LLM-based tool for BPMN collaboration modeling with multiple pools, lanes, message flows, and data objects. The tool supports a feedback loop, customizable prompts, adjustable LLM parameters, and the selection of supported BPMN constructs. Therefore, it provides a publicly available infrastructure for conducting user experiments to evaluate not only the capabilities of LLMs to generate process models but also to examine the impact of different usage patterns, prompting strategies, and user groups on the quality of the generated models.

Additionally, the integration of model checkers shifts the modeler's focus from domain-independent issues to the semantics of the model. This integration also enables a deeper analysis of recurring patterns and typical modeling errors generated by LLMs. Moreover, the tool opens opportunities to investigate user interaction patterns and how LLMs can leverage implicit knowledge to offer more effective and context-aware support throughout the modeling process.

Declaration on Generative AI

Generative AI was not used for preparing the text of this paper. The presented tool, as described in Sect. 2.1 uses generative AI for generating process models in a JSON format. However, the graphical representation of the generated JSON in the form of BPMN Collaboration diagrams, as shown in Figures 2, 3, and 4 is not based on generative AI.