
BPMN ASSISTANT: AN LLM-BASED APPROACH TO BUSINESS PROCESS MODELING

Josip Tomo Licardo
Faculty of Informatics
Juraj Dobrila University of Pula
Zagrebačka 30
52100 Pula, Croatia
jlicardo@unipu.hr

Nikola Tanković
Faculty of Informatics
Juraj Dobrila University of Pula
Zagrebačka 30
52100 Pula, Croatia
ntankov@unipu.hr

Darko Etinger
Faculty of Informatics
Juraj Dobrila University of Pula
Zagrebačka 30
52100 Pula, Croatia
detinger@unipu.hr

September 30, 2025

ABSTRACT

This paper presents BPMN Assistant, a tool that leverages Large Language Models (LLMs) for natural language-based creation and editing of BPMN diagrams. A specialized JSON-based representation is introduced as a structured alternative to the direct handling of XML to enhance the accuracy of process modifications. Process generation quality is evaluated using Graph Edit Distance (GED) and Relative Graph Edit Distance (RGED), while editing performance is evaluated with a binary success metric. Results show that JSON and XML achieve similar similarity scores in generation, but JSON offers greater reliability, faster processing, and significantly higher editing success rates. We discuss key trade-offs, limitations, and future improvements. The implementation is available at <https://github.com/jtlicardo/bpmn-assistant>.

1 Introduction

Business Process Model and Notation (BPMN) has long served as a standard for modeling business processes, enabling organizations to visualize and optimize their workflows. However, the complexity inherent in creating, editing, and interpreting BPMN diagrams poses significant challenges, particularly for individuals without specialized training [1]. As Recker [2] demonstrates through extensive research, BPMN's over-engineered nature and the prevalent lack of formal training among its users create substantial barriers to effective adoption. This bottleneck often results in inefficiencies and a reliance on experts, which can increase operational costs and delay decision-making.

A significant challenge in modern organizations is the communication gap between IT departments and business stakeholders. While IT professionals are comfortable with formal modeling notations and technical specifications, business users typically express processes in natural language and informal descriptions. A systematic review by [3] demonstrates that this disconnect leads to misunderstandings, requirements misalignment, and implementation delays, with communication barriers persisting despite significant investments in alignment initiatives. Furthermore, valuable process knowledge frequently remains trapped in the minds of domain experts or scattered across various informal documents, making it difficult to capture and formalize this expertise [4].

The complexity of extracting process models from unstructured text remains a significant challenge, as highlighted by Bellan et al.'s qualitative analysis [5]. Their research reveals that current methodologies primarily rely on ad-hoc rule-based approaches, which struggle with the complexity of real-world documents. This challenge is further compounded by the limitations of traditional process elicitation methods, as demonstrated by Baião et al. [6], who propose a novel approach integrating group storytelling techniques with text mining to capture the nuances of human-centric activities in process modeling.

Recent research has highlighted the cognitive challenges faced by modelers when creating process models, particularly regarding cognitive load and task complexity. Weber et al. [7] introduced a psycho-physiological approach to assess cognitive load during process modeling, utilizing real-time eye movement analysis to identify task-specific difficulties.

Their findings emphasize that process modeling requires substantial cognitive effort, especially in naming activities and managing complex structures. These insights reinforce the need for tools like BPMN Assistant, which aim to reduce cognitive barriers and enhance accessibility.

The technical barrier to BPMN usage, when combined with the dispersed nature of process documentation, creates a significant barrier to effective process management. Organizations often find themselves in a situation where valuable process knowledge exists but remains underutilized due to its informal and distributed nature.

Another critical issue is the dynamic nature of business processes in today's rapidly evolving business environment. Organizations need to frequently update and modify their processes to remain competitive and adapt to changing market conditions [8]. However, the formal nature of BPMN and the expertise required to modify process models often create a bottleneck in implementing these changes, leading to a gap between actual business operations and their formal documentation. This challenge is evidenced by Leopold et al.'s analysis of 585 BPMN models from industry, which revealed persistent quality issues related to model complexity, particularly in areas such as splits and joins, message flows, and model decomposition, highlighting the technical expertise required for effective BPMN modeling [9].

The disconnect between formal process models and actual business operations is further exacerbated by the increasing complexity of modern business processes. These processes often span multiple departments, involve numerous stakeholders, and integrate with various systems and external partners. As van der Aalst and Weijters [10] demonstrate in their seminal work on process mining, organizations face significant challenges in accurately capturing and analyzing these complex processes, including issues with hidden tasks, duplicate activities, and non-free-choice constructs. The traditional modeling approaches struggle to address these challenges while maintaining accessibility for all stakeholders.

Prior to the emergence of Large Language Models (LLMs), numerous attempts were made to automate aspects of process modeling and bridge the natural language gap. Early work by Friedrich et al. [11] demonstrated the potential of rule-based approaches for extracting process models from natural language text, though these systems struggled with complex sentence structures and domain-specific terminology. The integration of natural language processing into process management faced significant challenges with semantic ambiguity and contextual understanding. Efforts by Leopold et al. [12] to automatically generate natural language descriptions from process models highlighted both the potential and limitations of traditional NLP approaches in process automation. These early automation attempts, while groundbreaking, were often constrained by their reliance on predefined rules and limited ability to handle variations in natural language expressions [13].

Advancements in artificial intelligence, particularly the emergence of LLMs, have opened new avenues for automating the creation and management of business process models [14, 15, 16]. These models have demonstrated exceptional capabilities in understanding and generating natural language, making them well-suited for bridging the gap between textual process descriptions and formal BPMN representations. Recent research by Klievtsova et al. [17] has shown that AI-assisted approaches to process modeling can be particularly effective, with AI-generated models often being preferred over those created by inexperienced human modelers. However, Rebmann et al. [18] highlight that while LLMs show promise in process-related tasks, their effectiveness often depends on proper fine-tuning and task-specific training, particularly for complex semantics-aware operations. This suggests significant potential for LLM-based tools in democratizing access to process modeling capabilities, while also emphasizing the importance of careful implementation and training approaches.

This work presents BPMN Assistant, a system that leverages Large Language Models to address the fundamental challenges in business process modeling. While recent research has demonstrated the potential of AI-assisted process modeling, the critical aspect of model modification and maintenance remains largely unexplored. Our work advances the state of the art by demonstrating the effectiveness of LLMs in editing existing BPMN diagrams through natural language instructions.

Our evaluation reveals significant improvements in editing success rates compared to baseline approaches, particularly when using structured representations for model manipulation. These results suggest that LLMs can effectively bridge the gap between informal process descriptions and formal models, offering a promising solution for maintaining and updating business process documentation.

The following sections detail our methodology, system implementation, and evaluation results, providing a foundation for future research in AI-assisted process modeling.

2 Related Work

The integration of Large Language Models (LLMs) into Business Process Management (BPM) has led to significant developments in process automation and modeling. The introduction of specialized tools like the BPMN-Chatbot [19]

and ProMoAI [20] has demonstrated the practical applications of this technology. Initial explorations by Berti and Qafari [21] into using GPT-4 and Bard (Gemini) for process mining tasks showed promising results in interpreting both procedural and declarative models, highlighting the potential of LLMs in the BPM domain. These developments have led to comprehensive frameworks for process modeling with LLMs [22], showing promising results in both automation and model quality.

In the area of process extraction, various approaches have demonstrated different methodologies and results. Ferreira et al. [23] developed a semi-automatic method for identifying business process elements from natural language texts, achieving 91.92% accuracy in their prototype implementation through carefully defined mapping rules. Taking a different approach, Neuberger et al. [24] introduced a universal prompting strategy that leverages LLMs for process model information extraction, demonstrating performance improvements of up to 8% F1 score over traditional machine learning methods.

The integration of prompt engineering in BPM has emerged as a promising direction, as discussed by Busch et al. [25]. Their research highlights how prompt engineering can effectively utilize pre-trained language models without extensive fine-tuning, addressing common challenges in process extraction and predictive monitoring. This approach is particularly valuable in scenarios with limited data availability, offering a computationally sustainable solution for BPM applications.

ProMoAI, introduced by Kourani et al. [20], represents a significant development in LLM-driven process modeling. While the tool incorporates prompt engineering and error handling techniques, it currently generates Graphviz graph visualizations rather than process models based on standardized BPMN notation. The tool's effectiveness was thoroughly evaluated in a subsequent study [26], which demonstrated its capabilities when paired with different LLMs, with Claude 3.5 Sonnet emerging as the best-performing model for generating high-quality process models. Despite using non-standard notation, ProMoAI's approach demonstrates strong potential for using LLMs to assist in process visualization tasks.

Köpke and Safan [19] introduced the BPMN-Chatbot, a publicly available web-based tool designed for efficient LLM-based process modeling. The BPMN-Chatbot allows users to generate BPMN process models interactively using text or voice input. It achieved notable efficiency gains, reducing token usage by up to 94% compared to alternative tools like ProMoAI while maintaining a correctness rate of 95%, which surpassed the best competitor's 86%. The authors emphasized the importance of user testing to evaluate the feedback loop's capabilities, which are critical for interactive process design.

While BPMN-Chatbot demonstrates notable strengths, particularly in token efficiency and correctness rates, its evaluation primarily focuses on the generation of process models. The editing capabilities, which are essential for iterative refinement and interactive process modeling, have not been rigorously evaluated. Addressing this gap could further validate its utility in real-world scenarios where users frequently modify and adapt process models.

In a comprehensive evaluation of process extraction approaches, Bellan et al. [27] analyzed ten state-of-the-art methods for extracting process models from textual descriptions. Their systematic comparison revealed significant variations in performance and methodology among existing tools, with no single approach achieving superior results across all evaluation metrics. This study highlighted the need for standardized evaluation frameworks and emphasized the challenges in automated process extraction that our work aims to address.

Kourani et al. [22] proposed a comprehensive framework leveraging LLMs to automate the generation and refinement of process models from textual descriptions. This framework demonstrated its ability to streamline process modeling tasks while maintaining sound and executable model outputs. The superiority of this approach was evident in its comparison with traditional methods, particularly in resolving errors and integrating user feedback effectively. GPT-4 showcased strong performance in generating process models, addressing errors, and adapting to user feedback, whereas Gemini struggled with similar tasks.

Nivon et al. [28] introduced a novel approach to automating BPMN process generation from textual requirements, addressing the challenges faced by non-expert users in process modeling. Their methodology employs a three-step approach utilizing a fine-tuned GPT-3.5 model: first extracting tasks and ordering constraints from textual descriptions, then constructing an abstract syntax tree (AST) to represent task relationships, and finally converting the AST into a BPMN process. Their Java-based implementation achieved a 78.5% accuracy rate for valid BPMN processes in tests across 200 descriptions, demonstrating the viability of automated BPMN generation while highlighting areas for potential improvement through more advanced language models.

These studies collectively illustrate the transformative potential of LLMs in business process modeling. Tools like BPMN-Chatbot, ProMoAI, and Nivon et al.'s automated BPMN generator demonstrate the feasibility of combining generative AI with domain-specific methodologies to achieve higher efficiency, accuracy, and accessibility. Moreover,

the benchmarks and frameworks established by researchers provide a foundation for future advancements, emphasizing the importance of careful prompt design, user feedback integration, and domain-specific optimization in realizing the full potential of LLMs in BPM.

3 System Architecture

The BPMN Assistant system architecture, as illustrated in Figure 1, is designed to facilitate seamless interaction between users and BPMN models through natural language inputs. The system is composed of three primary components: the Python-based backend, the BPMN layout server, and the Vue.js frontend. Each component plays a critical role in ensuring the system's functionality, efficiency, and usability.

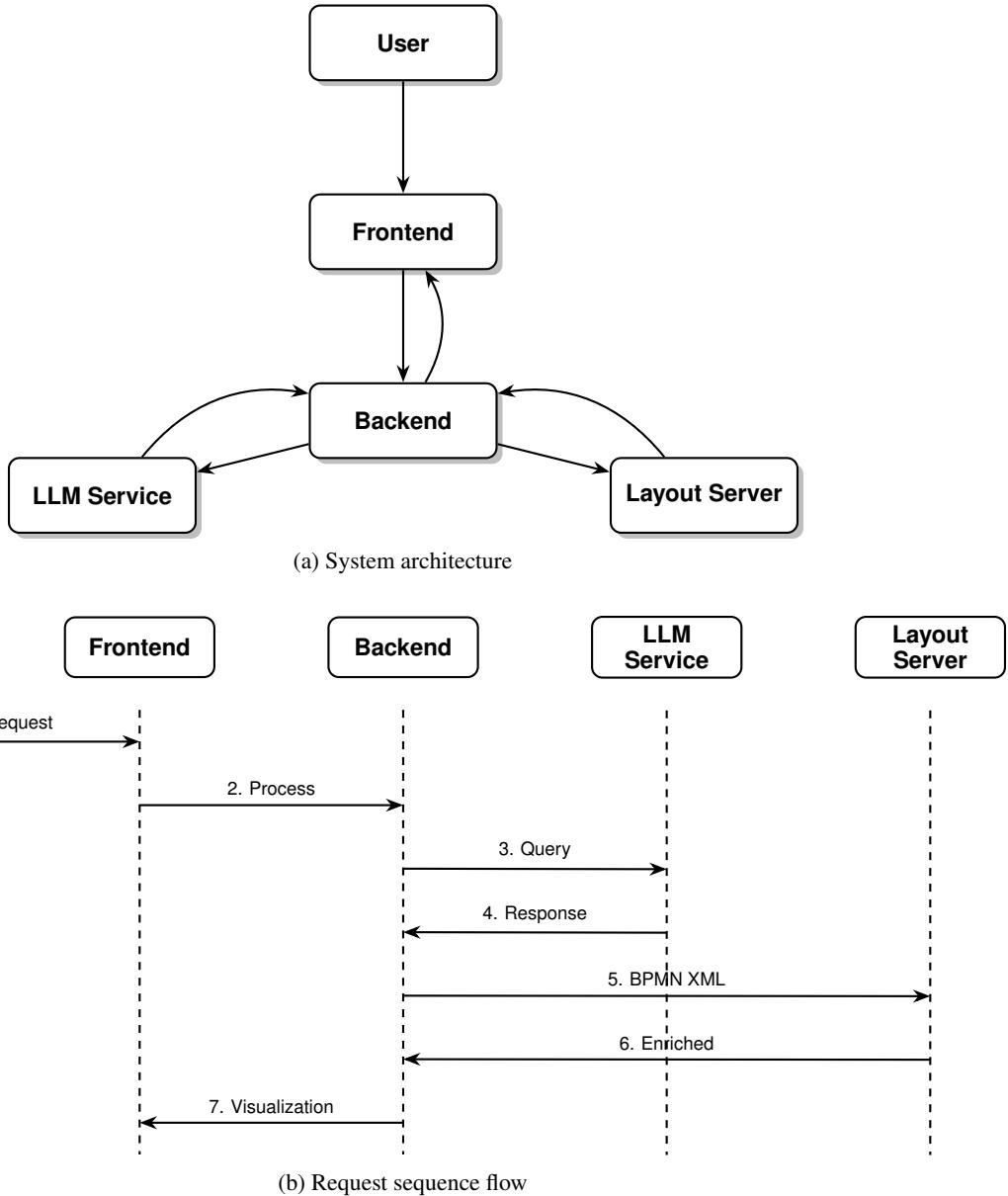


Figure 1: BPMN Assistant system: (a) Component architecture showing key system elements and (b) request sequence flow showing the temporal order of interactions.

3.1 Backend

The backend, implemented in Python, serves as the core computational engine of the system. Its primary responsibilities include handling user inputs, interacting with the Large Language Model (LLM), managing BPMN diagrams, and providing APIs for the frontend. The backend leverages the FastAPI framework, chosen for its performance and simplicity in developing RESTful APIs.

The backend serves as the first point of contact for user inputs, analyzing these inputs to determine the user's intent. This can include conversational queries, such as asking questions about BPMN concepts, or operational commands, such as creating or modifying BPMN diagrams. When a conversational intent is recognized, the backend communicates with the LLM to generate a natural language response. If the input requires an operational response, the backend constructs a prompt to instruct the LLM to generate or modify a BPMN diagram. These diagrams are initially represented in JSON format and subsequently converted to BPMN XML for compatibility with visualization tools. Additionally, the backend supports the uploading of BPMN files, enabling users to query or modify existing diagrams. The backend includes robust error handling and validation mechanisms to ensure that outputs from the LLM are accurate and reliable. If invalid JSON is generated, the backend retries the process or notifies the user of the error. By integrating with the BPMN layout server, the backend also ensures that diagrams are enriched with graphical information for enhanced visualization.

Model Integration and Selection Our system integrates multiple models from different providers, each chosen for its specific strengths and use cases. Table 1 provides an overview of the supported models and their capabilities.

Provider	Model	Description
OpenAI	GPT-4o	Multimodal model capable of processing text, audio, image, and video [29].
	GPT-4o mini	Fast and affordable model optimized for simpler tasks
	o3-mini	Cost-efficient reasoning model for coding and science use-cases.
Anthropic	Claude 3.5 Sonnet	Advanced model featuring a 200K token context window for enhanced reasoning.
Google	Gemini 2.0 Flash	Multimodal model with low latency and a 1M context window.
	Llama 3.3 70B	Meta's latest open-source LLM [30].
Fireworks AI	Qwen 2.5 72B	Alibaba's open-source model with strong multilingual capabilities [31].
	Deepseek V3	Open-source model specialized in technical and scientific reasoning [32].

Table 1: Overview of Integrated AI Models

Model	Input	Output
o3-mini	1.10	4.40
GPT-4o	2.50	10.00
GPT-4o mini	0.15	0.60
Claude 3.5 Sonnet	3.00	15.00
Gemini 2.0 Flash	0.10	0.40
Qwen 2.5 72B	0.90	0.90
Llama 3.3 70B	0.90	0.90
Deepseek V3	0.75	3.00

Table 2: Pricing for AI Models (in USD per million tokens)

This diverse selection ensures flexibility and performance optimization across different AI-driven tasks.

Supported BPMN Elements The backend facilitates operations on a variety of BPMN elements, including tasks, gateways, and events. These supported elements, detailed in Table 3, enable the system to cover a comprehensive range of BPMN functionalities, enhancing its applicability to diverse workflows.

Category	Elements
Tasks	General tasks, user tasks, service tasks
Gateways	Exclusive gateway, parallel gateway
Events	Start event, end event

Table 3: Supported BPMN Elements

3.2 BPMN Layout Server

The BPMN layout server, implemented using Node.js with the Express.js framework, is dedicated to augmenting BPMN diagrams with graphical information. This server employs the `bpmn-auto-layout` npm library to generate DI (Diagram Interchange) information, which includes the graphical coordinates of BPMN elements.

The layout server takes BPMN XML files from the backend and enhances them by adding graphical coordinates, which are essential for visual representation. Although the server efficiently enriches diagrams for most scenarios, it is currently limited in its ability to process multi-pool or multi-lane diagrams due to constraints in the `bpmn-auto-layout` library. The server is designed to work seamlessly with the backend, exchanging data through REST APIs to ensure smooth integration.

3.3 Frontend

The Vue.js-based frontend provides an intuitive graphical user interface (GUI) for user interaction. Its design focuses on accessibility, enabling non-experts to engage with BPMN modeling tasks effectively.

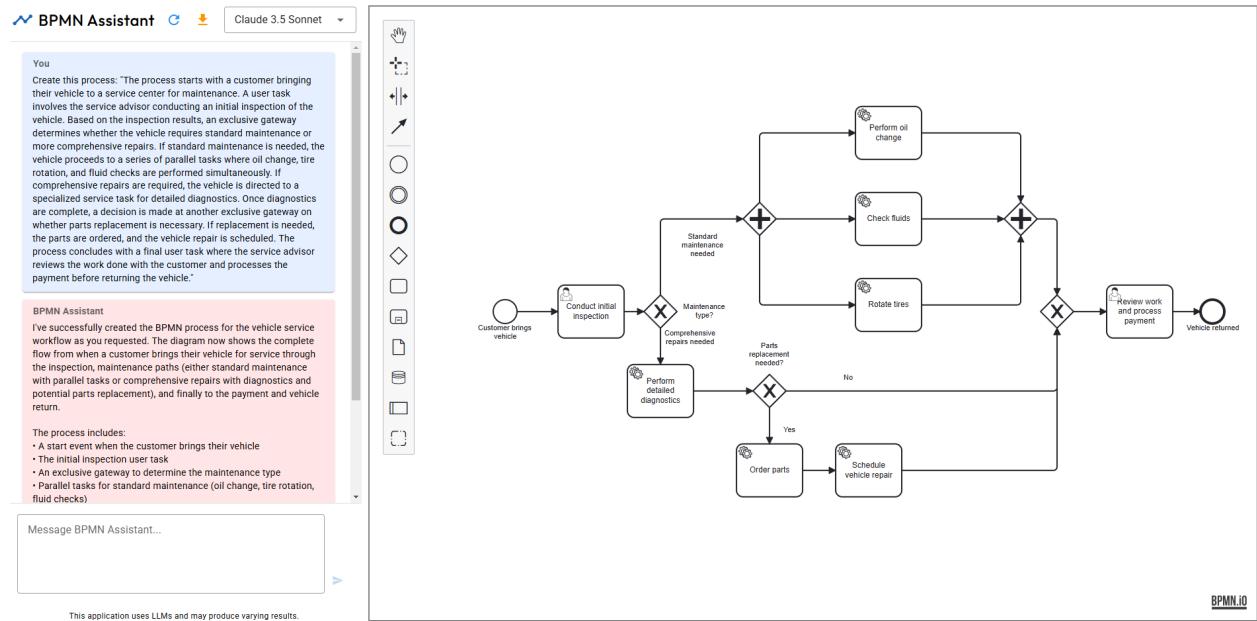


Figure 2: The web application interface featuring a dual-panel design: a chat interface on the left and a BPMN canvas on the right.

The frontend features a dual-panel design that facilitates interaction with both the LLM and BPMN diagrams. On the left, a chat interface allows users to submit natural language queries and view responses in real-time. Users can also select their preferred LLM for processing queries. On the right, a BPMN canvas, powered by `bpmn.io` (`bpmn.js`), displays the generated or modified BPMN diagrams. This canvas enables users to interact with the diagrams in a manner similar to traditional desktop tools. The frontend also provides feedback to users during diagram generation or modification, displaying status messages that indicate progress. Once a BPMN diagram is complete, users can download it for offline use or integration with other tools.

The frontend bridges the gap between users and the underlying system, providing a user-friendly interface for BPMN modeling and interaction.

3.4 Data Flow

The system processes data through a series of well-defined steps to ensure efficient operation and user satisfaction. Initially, user inputs are submitted via the frontend and transmitted to the backend, where intent recognition determines the appropriate response. If the input is conversational, the backend queries the LLM to generate a natural language reply. For operational inputs, the backend prompts the LLM to produce BPMN diagrams in JSON format. This JSON is then converted into BPMN XML and sent to the BPMN layout server. The layout server enriches the XML with graphical information and returns it to the backend. Finally, the enriched BPMN diagram is sent to the frontend, where it is displayed in the BPMN canvas. Users can choose to continue interacting with the diagram or download it for their records.

3.5 JSON Representation of BPMN Diagrams

The BPMN JSON representation uses a sequence of elements to describe the process. Each element is executed in order based on its position in the `process` array unless gateways (exclusive or parallel) specify branching paths. The following describes the structure of various BPMN elements supported by the system.

Tasks Tasks represent activities within a process and are specified with the `type` field. The system supports three task types: general tasks, user tasks, and service tasks. When defining tasks, it is recommended to specify the specific task type (`userTask` or `serviceTask`) when possible.

```
{
  "type": "task" | "userTask" | "serviceTask",
  "id": String,
  "label": String // short task description
}
```

Events Events mark the beginning or end of a process. The system supports two event types: `startEvent` and `endEvent`. These events are critical for defining the boundaries of a workflow.

```
{
  "type": "startEvent" | "endEvent",
  "id": String,
  "label": String // optional event label
}
```

Gateways Gateways determine the flow of a process by introducing branching or parallel paths. Two types of gateways are supported: exclusive and parallel gateways.

Exclusive gateways allow for conditional branching based on specified conditions. Each branch contains a condition and an optional path array of elements executed if the condition is met.

```
{
  "type": "exclusiveGateway",
  "id": String,
  "label": String, // gateway label
  "has_join": true | false, // whether the gateway merges branches
  "branches": [
    {
      "condition": String, // condition for the branch
      "path": [], // optional sequence of elements
      "next": String // optional next element ID
    }
  ]
}
```

Parallel gateways execute multiple branches simultaneously. Each branch is represented as an array of elements executed in parallel. Synchronization is automatically handled by the system, eliminating the need to define converging elements explicitly.

```
{
  "type": "parallelGateway",
  "id": String,
  "branches": [
    [], // branch 1 elements
    [] // branch 2 elements
  ]
}
```

This structured JSON representation serves as the backbone of the BPMN Assistant, allowing for the accurate and efficient translation of user inputs into process models.

For a more complete example of the BPMN JSON representation, please refer to the Appendix (Section A).

3.6 Process Editing Functions

The system supports a set of specialized functions for modifying BPMN diagrams. These functions, outlined in Table 4, enable precise control over process elements while maintaining the structural integrity of the diagram. When the LLM receives an editing request, it analyzes the natural language input and determines which function(s) to call to achieve the desired modifications.

Function	Parameters	Description
delete_element	element_id	Removes a specified element from the process
redirect_branch	branch_condition, next_id	Redirects a gateway branch to a new target element
add_element	element, before_id*, after_id*	Adds a new element at a specified position
move_element	element_id, before_id*, after_id*	Relocates an existing element within the process
update_element	new_element	Updates properties of an existing element

*Optional parameters, only one should be provided

Table 4: Process Editing Functions

The LLM processes editing requests through a structured approach by first analyzing the user's natural language request to understand the desired changes. It then identifies the affected elements in the current process and determines the appropriate editing functions required to achieve the requested modification. Once identified, the model generates the function calls with the correct parameters and ensures that the proposed changes maintain process integrity through validation.

Each editing function is designed to perform a specific type of modification while preserving the process's logical flow. For example, when deleting an element, the system automatically handles the reconnection of surrounding elements to maintain process continuity. Similarly, when adding new elements, the system ensures proper integration with existing process flows.

The granular nature of these functions allows the LLM to decompose complex editing requests into a series of atomic operations. This approach enhances reliability and makes it easier to validate and verify changes before they are applied to the process model.

4 Evaluation

The evaluation of BPMN Assistant focuses on assessing its accuracy through Graph Edit Distance (GED) and Relative Graph Edit Distance (RGED), two widely recognized metrics for measuring process model similarity [33, 34, 35]. Traditional GED methods often struggle with gateway semantics and execution probabilities, limiting their effectiveness in BPMN similarity measurements [36]. Schoknecht et al. [37] provide a comprehensive review of process model

similarity techniques, emphasizing that hybrid approaches—integrating syntactic, semantic, and behavioral comparisons—tend to produce more reliable results. However, many of these methods require domain-specific adaptations to be effective for BPMN.

Our evaluation methodology deliberately focuses on structural and syntactic assessment through graph similarity measures. This approach was selected based on the research objectives of comparing intermediate representations for BPMN generation and modification. It is important to note that this evaluation does not encompass semantic evaluation approaches such as those based on Petri nets or conformance checking from process mining. Similarly, we do not focus on execution performance through simulation studies, nor do we conduct comprehensive usability evaluations. While these aspects represent valuable dimensions for assessing process modeling tools, they fall outside the scope of the current research, which primarily examines the efficacy of different representation approaches in generating structurally accurate BPMN diagrams from natural language descriptions. The selected evaluation metrics allow for direct comparison between JSON and XML approaches while providing insight into the structural correctness of the generated models.

4.1 Graph Edit Distance (GED) and Relative Graph Edit Distance (RGED)

GED quantifies the cost of transforming one BPMN diagram into another by counting the minimum number of graph edit operations (e.g., node insertion, deletion, or substitution) required. It is a fundamental metric for evaluating the accuracy of diagram generation and modifications.

RGED normalizes the GED by considering the complexity of the involved graphs. This metric allows for a relative comparison of diagram similarity, regardless of their absolute sizes, providing a more interpretable evaluation score.

$$\text{Relative GED} = \frac{\text{GED}(G_1, G_2)}{\text{GED}(G_1, \text{Empty}) + \text{GED}(G_2, \text{Empty})} \quad (1)$$

$$\text{Similarity} = 1 - \text{Relative GED} \quad (2)$$

To implement GED, we utilized the NetworkX Python library, which provides efficient algorithms for computing graph edit distance. Unlike prior approaches that rely on domain-specific heuristics or ML-based enhancements [35], our implementation applies standard GED operations to measure similarity between BPMN diagrams.

While our evaluation focuses on structural similarity using GED and RGED, these metrics inherently reflect aspects of usability and clarity when compared against human-generated BPMN models. However, they do not explicitly capture higher-level qualitative factors such as readability, model simplicity, or semantic coherence, which are emphasized in studies like those by Huang and Kumar [38] and Pavlicek et al. [39].

4.2 Comparison with Baseline XML Editing

To establish the effectiveness of BPMN Assistant, we conducted a comparative analysis between our JSON-based BPMN representation and direct XML manipulation through language model prompting. The baseline method involves prompting the model to directly generate or modify BPMN XML specifications, which are then rendered into diagrams.

For this comparison, we selected a representative set of diagram modification tasks and executed them using both representations. We evaluated the approaches using RGED and its derived similarity score, as defined in Equations (1) and (2).

4.3 Generation Accuracy

The comparative analysis revealed that our JSON-based representation achieved an average similarity score of 0.70 compared to 0.69 for direct XML generation. While this indicates a slight numerical advantage for JSON, the difference is minimal, suggesting that both representations perform similarly in terms of structural accuracy. However, JSON demonstrated greater reliability, with fewer total failures across models.

Model	JSON	XML	Failures (JSON)	Failures (XML)
GPT-4o mini	0.67	0.67	0	1
GPT-4o	0.71	0.68	0	0
o3-mini	0.72	0.69	0	0
Claude 3.5 Sonnet	0.72	0.72	0	0
Gemini 2.0 Flash	0.70	0.66	0	0
Llama 3.3 70B Instruct	0.68	0.70	0	4
Qwen 2.5 72B Instruct	0.69	0.72	2	6
Deepseek V3	0.72	0.69	0	1

Table 5: Evaluation results showing similarity scores for different models using both our JSON-based BPMN representation and direct XML generation approaches. Each model was evaluated on 60 BPMN generation tasks.

Modality	Average Score	Total Failures
JSON	0.70	2
XML	0.69	12

Table 6: Average similarity scores and total failures per modality across all generated models.

Beyond similarity scores, JSON also outperformed XML in terms of efficiency. As shown in Table 7 and illustrated in Figure 3, JSON-based BPMN generation was significantly faster, with a mean latency of 14.41 seconds compared to 24.07 seconds for XML. Additionally, as depicted in Figure 4, JSON required more input tokens (2604.34 vs. 466.79 on average) but produced more concise outputs (849.29 vs. 1834.55 tokens). This trade-off suggests that JSON requires a richer input prompt but benefits from reduced processing time and smaller outputs, making it more efficient overall.

Metric	JSON	XML
Mean Latency (seconds)	14.41	24.07
Average Input Tokens	2604.34	466.79
Average Output Tokens	849.29	1834.55

Table 7: Summary of latency and token usage between JSON and XML representations for BPMN generation.

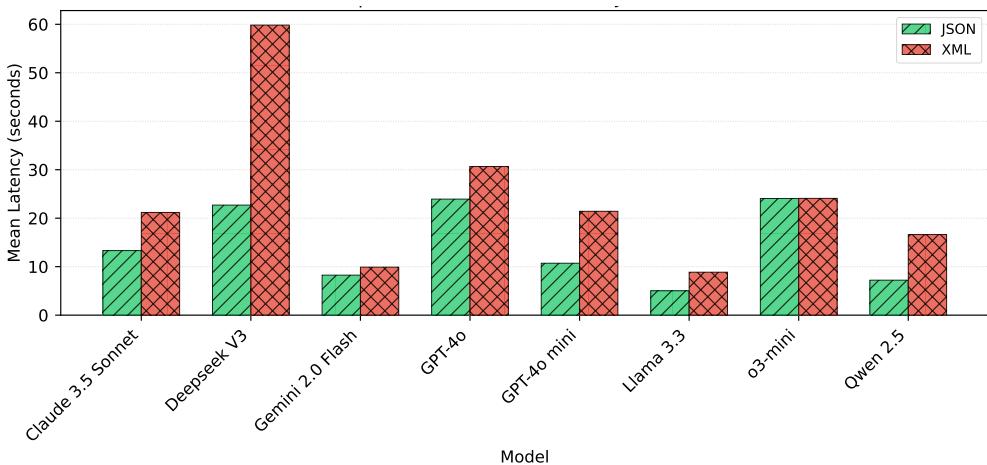


Figure 3: Latency comparison between JSON and XML-based BPMN generation.

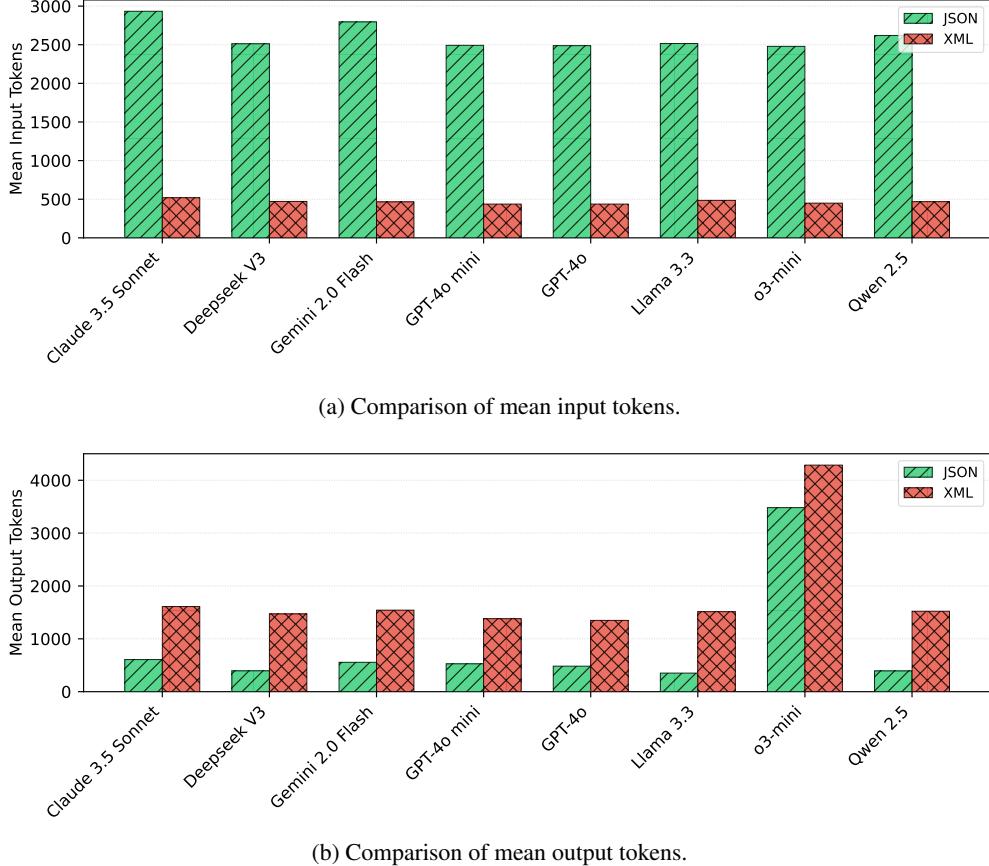


Figure 4: Token usage comparison for input and output tokens in JSON and XML-based BPMN generation. (a) shows the higher input token requirement for JSON prompts. (b) shows the significantly more concise output generated by the JSON approach.

4.4 Editing Capabilities

In addition to diagram generation, we evaluated the models' ability to interpret natural language editing instructions and perform the requested modifications. For this evaluation, we used a binary success/fail metric, as editing tasks involve correctly understanding the request and applying appropriate modifications to an existing diagram.

To determine validity, we implemented a Python-based validation function that checked whether the generated BPMN structure adhered to syntactic constraints. If a modification resulted in an invalid BPMN diagram, it was automatically marked as a failure. For valid outputs, an additional verification step assessed whether the requested edit had been correctly applied. This evaluation involved both automated checks and manual review to ensure accuracy.

The results, summarized in Table 8, show that models consistently achieved higher success rates when using our JSON-based approach compared to direct XML manipulation. JSON-based edits were more reliably executed across all tested models, likely due to the structured nature of JSON, which facilitates clearer representation of modifications.

Figure 5 provides a visual representation of these results, reinforcing the observed performance gap between JSON and XML approaches.

Our performance analysis also revealed substantial differences between JSON and XML approaches across both latency and token utilization metrics. As shown in Table 9, the JSON-based approach demonstrated significantly faster average processing times, requiring 21.46 seconds compared to 46.98 seconds for XML generation.

Token usage patterns showed an inverse relationship between input and output requirements for the two approaches. JSON-based generation demanded higher input token counts, averaging 19,567.70 tokens compared to XML's 5,102.74 tokens. However, this was offset by more compact outputs, with JSON generating an average of 364.14 tokens compared to XML's 2,665.32 tokens.

Model	JSON	XML
GPT-4o mini	0.45	0.05
GPT-4o	0.55	0.30
o3-mini	0.65	0.53
Claude 3.5 Sonnet	0.68	0.65
Gemini 2.0 Flash	0.45	0.33
Llama 3.3 70B Instruct	0.38	0.30
Qwen 2.5 72B Instruct	0.38	0.25
Deepseek V3	0.50	0.08

Table 8: Success rates for diagram editing based on natural language instructions, comparing our JSON-based approach with direct XML editing. Each model was evaluated on 40 diverse editing tasks.

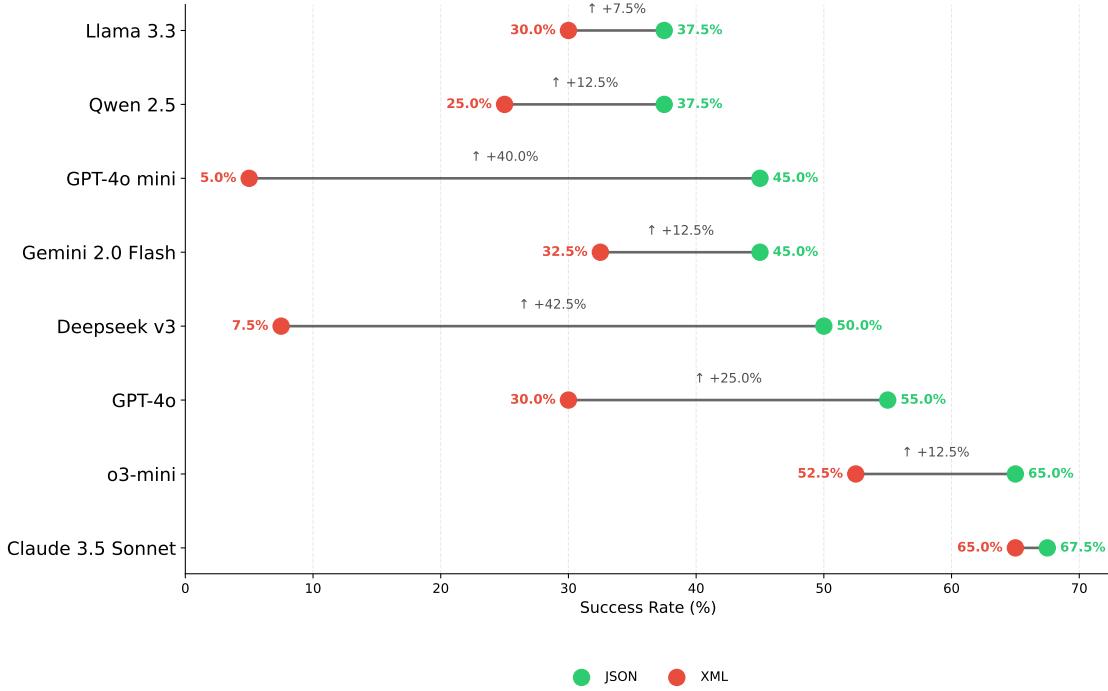


Figure 5: Success rate comparison for diagram editing tasks between JSON and XML-based BPMN representations.

Metric	JSON	XML
Average Latency (s)	21.46	46.98
Average Input Tokens	19,567.70	5,102.74
Average Output Tokens	364.14	2,665.32

Table 9: Comparative performance metrics between JSON and XML-based approaches for BPMN editing.

These performance characteristics suggest a clear trade-off between the approaches: while JSON requires more substantial input context, it achieves faster processing times and produces more concise outputs. The higher input token requirement of the JSON approach is offset by its reduced latency and more manageable output sizes, potentially offering better responsiveness in real-world BPMN editing scenarios. When combined with the higher editing success rates previously discussed, these performance metrics further support the advantages of the JSON-based approach for practical BPMN manipulation tasks.

5 Discussion

Our evaluation demonstrates that the JSON-based representation consistently outperforms direct XML manipulation across multiple metrics. Higher success rates in editing tasks (Table 8) suggest that structured intermediate representations are particularly valuable for complex model modifications. Performance differences between models indicate that model selection significantly impacts system effectiveness, with Claude 3.5 Sonnet and o3-mini showing particularly strong performance.

The inverse relationship between input and output token usage in the JSON versus XML approaches (Table 9) reveals interesting trade-offs that likely favor the JSON approach. Although the JSON approach requires more context in the prompts, it produces more concise output and faster processing times. This combination is particularly advantageous since input tokens typically cost less than output tokens, making the JSON approach potentially more cost-effective despite using more prompt tokens.

Importantly, BPMN Assistant is designed to make business process modeling more accessible to users without a technical background. Instead of requiring knowledge of specialized diagramming tools or technical formats like XML, users can simply describe a process in natural language. The system then automatically generates a professional diagram based on this description. This reduces the need for technical expertise, allowing business professionals to contribute directly to process design and updates, which can lead to more accurate and timely documentation.

The system’s ability to handle natural language editing commands has significant implications for process modeling workflows. The high success rates for editing tasks suggest that BPMN Assistant could reduce the technical barriers to process model maintenance, potentially leading to more accurate and up-to-date process documentation in practice.

Unlike previous tools such as BPMN-Chatbot [19] and ProMoAI [20], our system emphasizes model modification capabilities in addition to generation. A structured JSON representation offers advantages over both direct XML manipulation and alternative graph representations, such as the Graphviz DOT language¹, particularly in maintaining BPMN standard compliance and supporting complex editing operations.

Although our approach shows promise, several challenges remain. The dependency on LLM quality affects system reliability, and the current implementation’s limited support for BPMN elements restricts its application in complex scenarios. Additionally, the higher input token requirements of our JSON-based approach may impact costs in high-volume applications.

6 Limitations and Future Work

BPMN Assistant, while demonstrating promising results, has several limitations that should be acknowledged. A key limitation is the support for only a subset of BPMN elements (general tasks, user tasks, service tasks, exclusive gateways, parallel gateways, start events, and end events), which restricts the complexity of processes that can be modeled. However, since the focus of this work is on executable process models rather than collaboration diagrams, the absence of pools and lanes does not significantly impact our primary research objectives.

Due to the lack of semantic evaluation, the results should be considered experimental, as there is no guarantee that the business logic will be correctly integrated into the generated models. This limitation is particularly relevant when assessing the practical applicability of the generated diagrams in real-world scenarios.

The performance of BPMN Assistant is heavily dependent on the underlying LLM. Our evaluation revealed significant variations in performance across different models, with Claude 3.5 Sonnet and o3-mini showing particularly strong results. Further investigation into the specific capabilities and limitations of different LLMs in the context of process modeling would provide valuable insights for future improvements.

Human-computer interaction (HCI) studies would be necessary to properly evaluate the usability of the system for non-technical users. While BPMN Assistant aims to lower the barrier to process modeling through natural language interaction, comprehensive user studies would be required to validate this claim.

Another area of concern is the reliance on clear and unambiguous natural language input. Users might encounter difficulties in achieving this clarity, especially in multilingual or context-specific scenarios where language and terminology nuances can introduce ambiguity.

It is important to note that this work deliberately does not focus on semantic evaluation (e.g., using Petri nets) or conformance checking (e.g., process mining techniques), performance and execution time analysis (e.g., simulation studies), or comprehensive usability evaluation. The approach used in this research was selected considering the

¹<https://graphviz.org/doc/info/lang.html>

research objectives, the method of model generation, and existing evaluation methods cited in the literature. Future work could address these limitations by expanding the range of supported BPMN elements, incorporating semantic evaluation techniques, conducting usability studies, and optimizing the system for specific LLMs based on their strengths in process modeling tasks.

7 Conclusion

This paper presented BPMN Assistant, a novel approach to business process modeling that leverages Large Language Models to bridge the gap between natural language descriptions and formal BPMN representations. Through comprehensive evaluation, we demonstrated that our JSON-based representation for BPMN diagrams outperforms direct XML manipulation across multiple key metrics. In generation tasks, the JSON approach achieved lower latency with an average processing time of 14.41 seconds compared to 24.07 seconds for XML generation, while maintaining comparable model quality with an average similarity score of 0.70.

The evaluation results particularly highlight the effectiveness of our JSON-based representation in editing tasks, where it achieved consistently higher success rates across all tested models compared to XML-based editing, with editing operations taking an average of 21.46 seconds compared to 46.98 seconds for XML manipulation. While the JSON approach requires higher input token counts, this is offset by significantly more compact outputs, averaging 364.14 tokens compared to XML's 2,665.32 tokens. These findings suggest that structured intermediate representations can enhance the reliability and efficiency of LLM-based process modeling tools.

Despite these achievements, our work also reveals important areas for future research. The current limitations in supporting only a subset of BPMN elements indicate opportunities for expanding the system's capabilities, particularly for modeling more complex executable processes. Additionally, the varying performance across different LLMs suggests potential benefits from model-specific optimization approaches.

In broader terms, this research represents a significant step toward making business process modeling more accessible to a wider audience. By enabling natural language interaction with process modeling tools, BPMN Assistant reduces the technical knowledge required to create and modify business process diagrams. This accessibility has potential implications beyond technical efficiency gains, as it could foster better communication between technical and non-technical stakeholders in organizations. When business experts can directly interact with modeling tools using everyday language, they may become more engaged in the process modeling activities, potentially leading to more accurate process documentation and improved alignment between business needs and implemented processes. The natural language approach also opens possibilities for process modeling in educational contexts, where students can focus on understanding process concepts rather than mastering technical modeling interfaces.

Looking ahead, future work could focus on expanding support for additional BPMN elements to enable modeling of more complex executable processes. The significant performance variations observed across different LLMs also suggest opportunities for developing specialized prompt engineering techniques for process modeling tasks. These improvements would further enhance the practical applicability of LLM-based process modeling tools in enterprise settings.

References

- [1] Michael Rosemann. Potential pitfalls of process modeling: Part A. *Business Process Management Journal*, 12:377–384, May 2006.
- [2] Jan Recker. Opportunities and constraints: The current struggle with BPMN. *Business Process Management Journal*, 16(1):181–201, 2010.
- [3] Samgwa Quintine Njanka, Godavari Sandula, and Ricardo Colomo-Palacios. IT-Business Alignment: A Systematic Literature Review. *Procedia Computer Science*, 181:333–340, January 2021.
- [4] Han van der Aa, Henrik Leopold, Felix Mannhardt, and Hajo A. Reijers. On the Fragmentation of Process Information: Challenges, Solutions, and Outlook. In Khaled Gaaloul, Rainer Schmidt, Selmin Nurcan, Sérgio Guerreiro, and Qin Ma, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 3–18, Cham, 2015. Springer International Publishing.
- [5] Patrizio Bellan, M. Dragoni, and Chiara Ghidini. A Qualitative Analysis of the State of the Art in Process Extraction from Text. In *DP@AI*IA*, pages 19–30, 2020.
- [6] Fernanda Araujo Baião, Flávia Maria Santoro, and João Carlos de A. R. Gonçalves. Let Me Tell You a Story - On How to Build Process Models. *Journal of Universal Computer Science*, 2011.
- [7] Barbara Weber, Manuel Neurauter, Jakob Pinggera, Stefan Zugal, Marco Furtner, Markus Martini, and Pierre Sachse. Measuring Cognitive Load During Process Model Creation. In Fred D. Davis, René Riedl, Jan vom Brocke, Pierre-Majorique Léger, and Adriane B. Randolph, editors, *Information Systems and Neuroscience*, pages 129–136, Cham, 2015. Springer International Publishing.
- [8] Eskedar Gizat Desalegn, Maria João Coelho Guedes, Jorge Filipe Da Silva Gomes, and Shiferaw Mitiku Tebeka. Disentangling organizational agility from flexibility, adaptability, and versatility: A systematic review. *Future Business Journal*, 10(1):117, November 2024.
- [9] Henrik Leopold, Jan Mendling, and Oliver Günther. Learning from Quality Issues of BPMN Models from Industry. *IEEE Software*, 33(4):26–33, July 2016.
- [10] W.M.P. Van Der Aalst and A.J.M.M. Weijters. Process mining: A research agenda. *Computers in Industry*, 53(3):231–244, April 2004.
- [11] Fabian Friedrich, Jan Mendling, and Frank Puhlmann. Process Model Generation from Natural Language Text. In Haralambos Mouratidis and Colette Rolland, editors, *Advanced Information Systems Engineering*, pages 482–496, Berlin, Heidelberg, 2011. Springer.
- [12] Henrik Leopold, Jan Mendling, and Artem Polyvyanyy. Generating Natural Language Texts from Business Process Models. In Jolita Ralyté, Xavier Franch, Sjaak Brinkkemper, and Stanislaw Wrycza, editors, *Advanced Information Systems Engineering*, pages 64–79, Berlin, Heidelberg, 2012. Springer.
- [13] Jan Mendling, Henrik Leopold, and Lucinea Heloisa Thom. Natural Language Processing with Process Models (NLP4RE Report Paper). In *REFSQ Workshops 2019*, pages 1–5, 2019.
- [14] Michael Grohs, Luka Abb, Nourhan Elsayed, and Jana-Rebecca Rehse. Large Language Models Can Accomplish Business Process Management Tasks. In Jochen De Weerdt and Luise Pufahl, editors, *Business Process Management Workshops*, pages 453–465, Cham, 2024. Springer Nature Switzerland.
- [15] Timotheus Kampik, Christian Warmuth, Adrian Rebmann, Ron Agam, Lukas N. P. Egger, Andreas Gerber, Johannes Hoffart, Jonas Kolk, Philipp Herzig, Gero Decker, Han van der Aa, Artem Polyvyanyy, Stefanie Rinderle-Ma, Ingo Weber, and Matthias Weidlich. Large Process Models: Business Process Management in the Age of Generative AI. *KI – Künstliche Intelligenz*, 2024.
- [16] Ahmed Ali Linkon, Mujiba Shaima, Md Shohail Uddin Sarker, Badruddowza, Norun Nabi, Md Nasir Uddin Rana, Sandip Kumar Ghosh, Mohammad Anisur Rahman, Hammed Esa, and Faiaz Rahat Chowdhury. Advancements and Applications of Generative Artificial Intelligence and Large Language Models on Business Management: A Comprehensive Review. *Journal of Computer Science and Technology Studies*, 6(1):225–232, March 2024.
- [17] Nataliia Klievtsova, Janik-Vasily Benzin, Timotheus Kampik, Juergen Mangler, and Stefanie Rinderle-Ma. Conversational process modeling: Can generative AI empower domain experts in creating and redesigning process models? *arXiv preprint arXiv:2304.11065*, 2023.
- [18] Adrian Rebmann, Fabian David Schmidt, Goran Glavaš, and Han Van Der Aa. Evaluating the Ability of LLMs to Solve Semantics-Aware Process Mining Tasks. *2024 6th International Conference on Process Mining (ICPM)*, pages 9–16, October 2024.

- [19] Julius Köpke and Aya Safan. Introducing the BPMN-Chatbot for Efficient LLM-Based Process Modeling. In *International Conference on Business Process Management*, pages 1–5, 2024.
- [20] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M.P. Van Der Aalst. ProMoAI: Process Modeling with Generative AI. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8708–8712, Jeju, South Korea, August 2024. International Joint Conferences on Artificial Intelligence Organization.
- [21] Alessandro Berti and Mahnaz Sadat Qafari. Leveraging Large Language Models (LLMs) for Process Mining (Technical Report), July 2023.
- [22] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. Process Modeling With Large Language Models. In *Enterprise, Business-Process and Information Systems Modeling*, volume 511, pages 229–244. Springer, Cham, 2024.
- [23] Renato César Borges Ferreira, Lucinéia Heloisa Thom, and Marcelo Fantinato. A Semi-automatic Approach to Identify Business Process Elements in Natural Language Texts:. In *Proceedings of the 19th International Conference on Enterprise Information Systems*, pages 250–261, Porto, Portugal, 2017. SCITEPRESS - Science and Technology Publications.
- [24] Julian Neuberger, Lars Ackermann, Han van der Aa, and Stefan Jablonski. A Universal Prompting Strategy for Extracting Process Model Information from Natural Language Text using Large Language Models. In *Conceptual Modeling: 43rd International Conference, ER 2024*, pages 38–55, 2025.
- [25] Kiran Busch, Alexander Rochlitzer, Diana Sola, and Henrik Leopold. Just Tell Me: Prompt Engineering in Business Process Management, April 2023.
- [26] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. Evaluating Large Language Models on Business Process Modeling: Framework, Benchmark, and Self-Improvement Analysis. *Software and Systems Modeling*, 2024.
- [27] Patrizio Bellan, Mauro Dragoni, Chiara Ghidini, Han van der Aa, and Simone Paolo Ponzetto. Process Extraction from Text: Benchmarking the State of the Art and Paving the Way for Future Challenges, October 2023.
- [28] Quentin Nivon and Gwen Salaün. Automated Generation of BPMN Processes from Textual Requirements. In Walid Gaaloul, Michael Sheng, Qi Yu, and Sami Yangui, editors, *Service-Oriented Computing*, pages 185–201, Singapore, 2025. Springer Nature.
- [29] OpenAI et al. GPT-4o System Card, October 2024.
- [30] Aaron Grattafiori and et al. The Llama 3 Herd of Models, November 2024.
- [31] Qwen et al. Qwen2.5 Technical Report, January 2025.
- [32] DeepSeek-AI et al. DeepSeek-V3 Technical Report, December 2024.
- [33] Josip Tomo Licardo, Nikola Tanković, and Darko Etinger. A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing. *Procedia Computer Science*, 239:483–490, January 2024.
- [34] Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors, *Business Process Management*, volume 5701, pages 48–63, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
- [35] Abid Sohail, Ammar Haseeb, Mobashar Rehman, Dhanapal Durai Dominic, and Muhammad Arif Butt. An Intelligent Graph Edit Distance-Based Approach for Finding Business Process Similarities. *Computers, Materials & Continua*, 69(3):3603–3618, 2021.
- [36] Indra Waspada and Rivanarto Sarno. An Improved Method of Graph Edit Distance for Business Process Model Similarity Measurement. In *2020 4th International Conference on Informatics and Computational Sciences (ICICoS)*, pages 1–6, November 2020.
- [37] Andreas Schoknecht, Tom Thaler, Peter Fettke, Andreas Oberweis, and Ralf Laue. Similarity of Business Process Models—A State-of-the-Art Analysis. *ACM Computing Surveys*, 50(4):1–33, July 2018.
- [38] Zan Huang and Akhil Kumar. New Quality Metrics for Evaluating Process Models. In Danilo Ardagna, Massimo Mecella, and Jian Yang, editors, *Business Process Management Workshops*, volume 17, pages 164–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [39] Josef Pavlicek, Radek Hronza, Petra Pavlickova, and Klara Jelinkova. The Business Process Model Quality Metrics. In Robert Pergl, Russell Lock, Eduard Babkin, and Martin Molhanec, editors, *Enterprise and Organizational Modeling and Simulation*, volume 298, pages 134–148, Cham, 2017. Springer International Publishing.

A Appendix: Process Example

The following is an example process modeled using the BPMN Assistant based on the textual description:

Textual description:

"The manager sends the mail to the supplier and prepares the documents. At the same time, the customer searches for the goods and picks up the goods."

JSON Representation:

```
{
  "process": [
    {
      "type": "startEvent",
      "id": "start",
    },
    {
      "type": "parallelGateway",
      "id": "parallel1",
      "branches": [
        [
          {
            "type": "serviceTask",
            "id": "task1",
            "label": "Send mail to supplier",
          },
          {
            "type": "task",
            "id": "task2",
            "label": "Prepare the documents",
          }
        ],
        [
          {
            "type": "task",
            "id": "task3",
            "label": "Search for the goods",
          },
          {
            "type": "task",
            "id": "task4",
            "label": "Pick up the goods",
          }
        ]
      ],
      {
        "type": "endEvent",
        "id": "end1",
      }
    ]
  }
}
```