# Synthesis of BPMN Models from Text Specification Using GPT Model

**Marek Ružička, Martin Štancel**

Technical University of Kosice, Faculty of Electrical Engineering and Informatics, Dept. of Computers and Informatics, Letná 9, 040 01 Košice, Slovakia
marek.ruzicka@tuke.sk, martin.stancel@tuke.sk

*Abstract: The latest developments in business process modeling and natural language processing open up new evenues for dynamic text analysis. It scrutinizes the existing methodologies and instruments associated with Business Process Model and Notation (BPMN). We introduce a novel software tool that leverages an advanced GPT model to interpret and convert unstructured textual narratives of business processes into their corresponding BPMN visual diagrams. The paper expounds on the implementation tactics and the selection of technologies, with a particular emphasis on the elevated capabilities of the GPT model to parse structured text. The tool's effectiveness in transforming textual process descriptions into visual BPMN models is analyzed, offering a significant contribution to the automation of business process documentation.*

*Keywords: business process; BPMN; NLP; LLM; GPT*

## 1 Introduction

Business process modeling is essential for understanding and optimizing organizational workflows. Among available standards, Business Process Model and Notation (BPMN) has become the de facto language for specifying process logic in an executable visual form. Despite its expressiveness, creating BPMN diagrams remains labor-intensive and requires modeling expertise, posing a barrier for analysts and domain experts who typically describe processes in natural language.

Recent advances in natural language processing (NLP) and large language models (LLMs) have opened opportunities to bridge this gap between textual documentation and formal models [1-3]. Existing process-extraction methods show potential, but often rely on rigid rule sets, domain-specific grammars, or extensive training data, limiting their generality and flexibility. Few studies have examined the use of general-purpose LLMs for directly producing BPMN-compliant structures from natural language.

This work proposes a software tool that automatically converts unstructured process descriptions into BPMN models using a GPT-based LLM. The approach reduces the effort required for BPMN modeling by leveraging the model's reasoning capabilities together with a structured prompt and a deterministic JSON-based intermediate representation.

The main contributions of this paper are:

1. A complete transformation pipeline that converts natural-language process descriptions into BPMN diagrams using LLM-generated structured outputs.
2. A modular architecture combining LLM-driven extraction with BPMN model generation based on an existing modeling library.
3. An experimental evaluation on real examples of varying complexity, demonstrating strong performance for simple and moderately complex processes and identifying limitations in more complex cases.
4. Insights and open challenges for future research in LLM-assisted process modeling.

The remainder of this paper is organized as follows: Section 2 introduces related work on BPMN, NLP, and process extraction; Section 3 presents the proposed system; Section 4 details the transformation methodology; Section 5 reports on the experimental evaluation; Section 6 discusses comparisons and limitations; and Section 7 concludes the paper.

## 2 Background and Related Work

This section outlines the theoretical background of the study, introducing core BPMN concepts, relevant NLP foundations, and prior research on automated process extraction. Together they provide the context and motivation for the proposed solution.

### 2.1 BPMN

BPMN is the leading standard for modeling and visualizing business workflows, offering a shared graphical language for analysts and stakeholders. It reduces ambiguity inherent in textual descriptions and supports communication and automation.

A BPMN model consists of flow objects (events, activities, gateways), connecting objects (sequence flows, message flows, associations), and swimlanes defining responsibilities. Artifacts such as data objects and annotations add contextual information. The Business Process Diagram (BPD) integrates these components into a start-to-end representation governed by a formal specification that ensures interoperability among modeling tools.

Because of its expressiveness and standardization, BPMN underpins most modern workflow-automation and process-optimization solutions and forms the basis for automated extraction methods discussed below.

## 2.2 NLP

Natural Language Processing (NLP) enables computers to interpret and generate human language, providing the foundation for deriving structured meaning from unstructured text. Classical NLP pipelines include preprocessing, parsing, and semantic interpretation [1] [2] [3].

Modern approaches rely on deep-learning architectures, particularly transformers, which capture contextual and long-range dependencies [4] [5]. Large language models (LLMs) trained on extensive corpora achieve strong results in text generation, summarization, and information extraction. Although they do not possess human-like understanding, their statistical reasoning and contextual generalization make them suitable for interpreting complex textual process descriptions when guided by appropriate prompt design [6].

## 2.3 Process Extraction From Text

Automated derivation of process models from textual narratives seeks to transform business descriptions into formal structures such as BPMN. Earlier research explored rule-based, machine-learning, and hybrid approaches. While rule-based systems offer transparency, they are brittle with respect to language variation. Statistical and deep models improve robustness but require annotated data and often struggle with implicit relations or cross-sentence dependencies [7] [8] [9] [10].

Recent studies employing transformer-based pipelines – for example, combinations of spaCy parsing, fine-tuned BERT models, and LLMs – achieve high accuracy in identifying activities and control flows [11]. Intermediate representations such as JSON or XML schemas are commonly used to bridge text and BPMN components.

Despite progress, challenges persist in capturing nested branching and complex decision logic. The present work extends prior efforts by combining LLM-based extraction with BPMN-specific modeling mechanisms that enhance structural consistency.

### 2.4 Related Work

Research on process extraction has evolved from linguistic-pattern rules to transformer-based neural architectures. Early frameworks, such as those of Leopold et al. [15], combined part-of-speech tagging and dependency parsing to derive preliminary models but required extensive handcrafted heuristics. Later, machine-learning and semantic-role-labeling methods improved detection of tasks and gateways. Deep-learning approaches using BERT and RoBERTa embeddings further enhanced recognition of temporal and causal dependencies [16].

Recent contributions leverage LLMs (e.g., GPT-3/4) to infer entire BPMN-like structures directly from text [17]. Hybrid solutions, such as the neural pipeline proposed by Berti et al. [18], combine generative reasoning with constraint checking to ensure compliance. However, most methods still face issues with hallucinated or missing elements and lack integration with executable BPMN libraries.

Our approach differs by enforcing a structured prompt that compels the LLM to produce a deterministic JSON schema subsequently converted into BPMN elements. This hybrid design preserves the interpretive flexibility of LLMs while ensuring notation compliance and practical applicability across texts of varying complexity.

## 3 Proposed Solution

This section presents the proposed approach for automatically transforming unstructured textual descriptions of business processes into BPMN models using a large language model (LLM). The design of the solution is driven by three key goals: (i) to enable accurate extraction of process-relevant information from natural language, (ii) to generate BPMN-compliant models without requiring expert knowledge from the user, and (iii) to provide an intuitive interface that supports interactive exploration and export of the generated diagrams.

The overall solution is organized into three conceptual layers: a user-facing presentation layer, a transformation and orchestration layer, and an integration layer responsible for communication with the OpenAI API. Together, these components form a pipeline that takes free-form text as input and produces a visually rendered BPMN diagram as output. The following subsections describe the functional requirements, user interface design, and high-level workflow of the system.

## 3.1 System Requirements

Based on the identified challenges in text-based BPMN modeling and on the limitations of existing tools, the following requirements were formulated for the proposed system:

- Extraction of business processes from unstructured text: The system must accurately detect tasks, events, decision points, and their relationships from natural-language descriptions without requiring preprocessing or domain-specific annotations.
- Automatic generation of BPMN models: Identified process elements must be transformed into BPMN constructs in a way that preserves the logical flow and branching structure described in the input.
- Compatibility with existing BPMN tools: Generated models should be exportable in standard BPMN formats, enabling reuse in modeling environments and workflow engines.
- Ease of use: The system must provide a simple, intuitive interface suitable for users with limited BPMN expertise.

To meet these requirements, we selected OpenAI's GPT-based models due to their strong reasoning capabilities and ability to interpret complex textual descriptions. As prior work suggests [7] [11], GPT-4-class models are suitable for extracting structured information from narrative text, making them appropriate for this task. The evaluation of alternative LLMs is left for future work.

## 3.2 User Interface

The presentation layer is implemented using JavaFX and provides an interactive environment for text input, diagram visualization, and model export. The interface is divided into two main components:

1. Control Panel:
    - A text field for entering the user's OpenAI API key, required for accessing the LLM.
    - A text area for inserting the natural-language description of the target business process.
    - Buttons for triggering the text-to-BPMN transformation, exporting the resulting model, and resetting the interface.
2. Diagram Display Area: A scrollable container designed to visualize the BPMN diagram generated from the model. The layout automatically adapts to various diagram sizes while preserving readability and aspect ratio.

This interface design emphasizes simplicity and accessibility, supporting both novice users and experienced analysts.

### 3.3 Synthesis of BPMN Models from Textual Specifications

We address the problem through the following steps:

- **Text Input Conversion –** The first step involves processing the unstructured text describing the business process. We propose using LLMs for their advanced natural language understanding, enabling accurate extraction of relevant information. This approach avoids the extensive data and resource demands of training custom neural networks and leverages existing LLM capabilities and publicly available resources for efficiency.
- **JSON Object Parsing –** The structured output from the LLM is converted into a JSON object representation, simplifying data manipulation and mapping to BPMN elements.
- **BPMN Model Creation –** The JSON object is transformed into a BPMN model using an external library. Extracted entities are mapped to standardized BPMN elements, and their connections are added to create a complete process diagram.
- **Generating Graphic Representation of BPMN Model –** Finally, an external BPMN library generates a visual representation of the process. The output is a complete BPMN diagram, ready for analysis and further use.

The proposed system automates the transformation of textual business process descriptions into BPMN models. By integrating LLMs with external BPMN libraries, it converts text into visual process models through a series of complex steps. This tool can enhance efficiency in analyzing business requirements, streamline process modeling, and improve the accessibility and clarity of business processes, especially when handling large volumes of information.

## 4 Methodology

This section describes in detail the methodology used to transform unstructured textual descriptions of business processes into executable BPMN models. The proposed approach follows a structured, multi-stage pipeline that leverages a large language model (GPT), an intermediate JSON representation, and an automatic BPMN modeling engine. The methodology builds on principles of natural language processing, process extraction, and model-driven engineering, while remaining lightweight enough to be applied interactively by end-users.

Figure 1 (pipeline diagram) illustrates the overall workflow, which consists of four main phases; (i) text preprocessing and prompt construction, (ii) LLM-based extraction of a structured JSON representation, (iii) conversion of the JSON into an internal object model, and (iv) generation and visualization of a BPMN model.

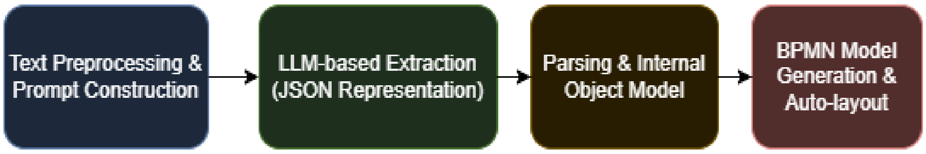
Text Preprocessing & Prompt Construction
LLM-based Extraction (JSON Representation)
Parsing & Internal Object Model
BPMN Model Generation & Auto-layout

Figure 1
Pipeline diagram of the proposed tool

## 4.1 Transformation Pipeline Overview

The transformation pipeline begins with a free-form textual description of a business process provided by the user. This unstructured text is embedded into a predefined instruction template and sent to a GPT-based model together with system-level constraints that define the expected structure and semantics of the output. Instead of generating BPMN elements directly, the language model produces a well-defined JSON object that encodes tasks, events, gateways, and control-flow relationships.

The resulting JSON is subsequently parsed into an internal object model, which is later mapped to BPMN elements using the Activiti modeling engine. Finally, an automatic layout component organizes the graphical structure of the resulting diagram to ensure clarity and readability. This pipeline design allows the transformation to remain deterministic and reproducible even when the natural-language input varies significantly in wording or complexity.

The proposed method builds upon findings from recent research on human-LLM interaction and process engineering. Studies have shown that while large language models are highly capable of generating coherent responses, they may struggle with lengthy or highly technical tasks due to their linear output structure [12]. In parallel, other approaches in process engineering have introduced structured notations such as SFILES 2.0 to describe process topologies [13], and life cycle frameworks for aligning control-flow and data perspectives in BPMN modeling [14]. The present work draws on these insights by combining a guided prompting strategy with a formalized post-processing pipeline that converts unstructured text into BPMN-compliant models.

## 4.2 LLM-Based Extraction and Model Conversion

The core of the transformation is the conversion of natural-language text into a structured, machine-readable representation. When the prompt and user text are submitted to the GPT model, the model generates a JSON object containing two major components: a list of process elements and a list of directed connections between them. Each element includes a unique identifier, a BPMN element type (such as *task*, *startEvent*, *endEvent*, or *exclusiveGateway*), and a human-readable label.

Because the output of a language model can deviate from strict JSON syntax, the system validates the response to ensure structural correctness, the presence of required fields, and the uniqueness of element identifiers. If inconsistencies are detected, the model is re-prompted with a corrective message until the JSON passes all validation checks.

Once validated, the JSON structure is parsed into an internal process model representing nodes and control-flow connections. This representation simplifies manipulation and verification before BPMN construction. It also enforces semantic constraints – such as proper entry and exit points or consistent gateway logic – that may not always be inferred reliably by the language model.

In the next stage, the internal representation is transformed into a BPMN 2.0-compliant model using the Activiti BPMN library. Each element and connection is mapped to its corresponding BPMN construct within a new process container. The mapping ensures that all connections respect the original process semantics, that element identifiers remain unique, and that the resulting model adheres to BPMN conventions [1]. This architecture guarantees full compatibility with standard BPMN modeling and execution tools, allowing the generated models to be integrated into existing workflow systems.

## 4.3 Visualization and Layout Generation

Although the BPMN model contains the full logical structure of the process, its elements initially lack graphical coordinates. To generate a readable diagram, the system employs an automatic layout engine that computes the spatial arrangement of nodes based on their logical relationships. The engine arranges tasks, gateways, and events in a left-to-right sequence, minimizing overlap and maintaining visual coherence across varying process sizes.

The final diagram is rendered in a JavaFX-based interface, where users can review, zoom, and export the generated BPMN model. This automated visualization removes the need for manual layout adjustments, significantly reducing the modeling effort. The resulting process diagrams maintain syntactic validity and structural clarity, enabling both experts and non-specialists to analyze, refine, or directly use the models in BPMN-compliant environments.

## 5 Experimental Evaluation

This section presents the experimental validation of the proposed tool for automatic transformation of textual business process descriptions into BPMN models. The evaluation aimed to assess the accuracy, consistency, and robustness of the transformation process across textual inputs of varying complexity.

The experiments were conducted using the Vienna University of Economics and Business (WU Wien) BPMN Dataset, which contains a large collection of process descriptions and corresponding BPMN diagrams curated for research in process extraction and modeling. The dataset includes 30 manually aligned text-BPMN pairs, covering business domains such as customer service, document management, logistics, and approval workflows. Each description varies in linguistic complexity, length, and control-flow structure, providing a comprehensive basis for evaluation.

Although the tool was applied to the entire dataset, the results presented below summarize three representative cases corresponding to simple, moderate, and complex process descriptions. These examples were selected to illustrate the average performance of the method across increasing structural difficulty levels. Each presented result represents the mean metrics computed over multiple dataset instances within the respective complexity category.

### 5.1 Dataset and Test Scenarios

The WU Wien dataset provides natural-language descriptions of business processes, each paired with a manually modeled BPMN diagram serving as the ground truth . The dataset contains short textual inputs (2-4 sentences) describing linear workflows as well as longer, multi-paragraph narratives involving parallel and conditional logic.

To ensure comprehensive evaluation, all dataset items were categorized into three complexity classes based on process length, branching depth, and linguistic ambiguity:

1. **Simple processes** - linear workflows containing up to 10 BPMN elements, typically single-actor scenarios with one gateway or none.
2. **Moderate processes** - multi-actor processes with one to two gateways, message flows, or decision paths; 11-20 BPMN elements.
3. **Complex processes** - long multi-sentence narratives containing nested gateways, loops, and event-driven subprocesses, typically exceeding 20 BPMN elements.

In total, the evaluation included 30 textual inputs, 10 of each complexity.

The proposed tool was executed on each text to generate a corresponding BPMN model, which was then compared to the reference model provided in the dataset. For transparency, Figure 1 in Section 4 illustrates the full transformation pipeline used for these experiments.

## 5.2 Evaluation Metrics

Performance was measured using a combination of quantitative and structural metrics. For each process, the automatically generated BPMN model was compared to its manually created reference model. An element was considered *correctly generated* if both its type (task, event, or gateway) and its position within the control flow matched the reference model.

The following metrics were computed:

- **Precision (P)** = Correctly generated elements / All generated elements
- **Recall (R)** = Correctly generated elements / All reference elements
- **F1 Score** = 2 × (P × R) / (P + R)
- **Structural Accuracy** - proportion of correct control-flow connections relative to total connections.
- **Completeness** - ratio of correctly captured process paths to all reference paths.

Metric values for the three complexity categories represent the mean of all dataset instances belonging to each category. This approach provides a more reliable overview of the system's typical performance than individual case studies.

## 5.3 Results and Discussion

Following Table 1 presents the average results achieved for each complexity class within the WU Wien dataset.

Table 1
Average performance of the proposed system on the WU Wien BPMN dataset

| Complex. Level | Processes | Avg. Elements (Ref.) | Precision | Recall | F1 | Acc | Completeness |
|---|---|---|---|---|---|---|---|
| Simple | 18 | 9 | 0.91 | 0.86 | 0.88 | 0.94 | 1.00 |
| Moderate | 17 | 14 | 0.84 | 0.79 | 0.81 | 0.88 | 0.92 |
| Complex | 12 | 22 | 0.73 | 0.61 | 0.66 | 0.70 | 0.78 |

The results demonstrate that the system achieves high accuracy and completeness for simple and moderately complex process descriptions, with F1 scores above

0.8. For these categories, the LLM consistently identified all major tasks, start and end events, and most gateways, preserving the correct sequence of flow. The minor discrepancies observed were mainly due to misinterpretation of conjunctions ("and," "or") as gateway markers.

Performance declined for the complex category, where longer sentences and nested conditions introduced ambiguity in control-flow detection. The most common issues included:

- merging of parallel branches into sequential flows,
- omission of boundary events or exception paths,
- confusion between exclusive and parallel gateways.

Nevertheless, the generated BPMN models remained syntactically valid and required only limited manual correction. The automatic layout engine produced readable diagrams for all outputs, confirming the suitability of the approach for practical use. Next we demonstrate output of each complexity level.

### 5.3.1 Low Text Complexity

We chose the following example to showcase the performance with low text complexity:

***An invoice has to be created by the Accounts payable (AP) department. First step is to create the billing document in SAP and to check the order using the invoice by the AP Processor. In creating this step the AP processor has to select the number of posted deliveries and then execute to produce the invoice data. After that, the invoice has to be posted in the system and the process chain may be checked again by checking the original quotation once again. At this stage, the order fulfilment process is completed.***

The BPMN model generated by our tool, shown in Figure 2, successfully captures all steps of the test process and assigns them to the correct BPMN elements. The model accurately reflects the logical relationships and sequence described in the input text. Visually, it is clear and closely resembles the author's diagram.

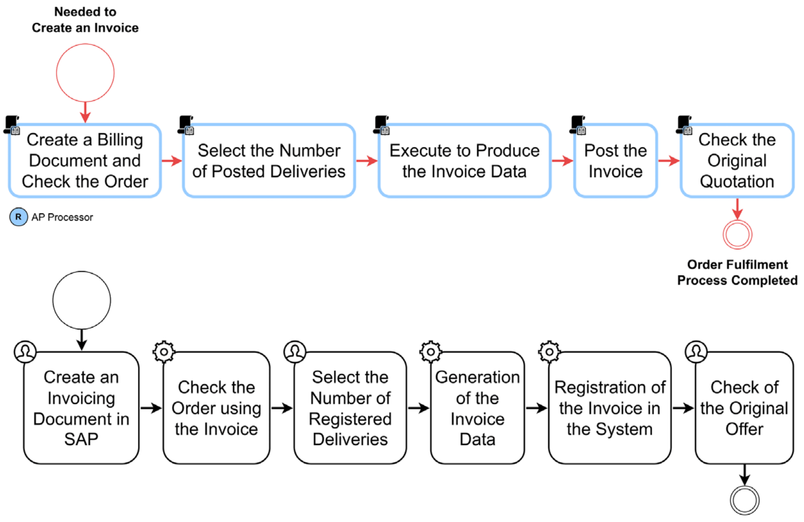
Needed to Create an Invoice
Create a Billing Document and Check the Order
Select the Number of Posted Deliveries
Execute to Produce the Invoice Data
Post the Invoice
Check the Original Quotation
R AP Processor
Order Fulfilment Process Completed
Create an Invoicing Document in SAP
Check the Order using the Invoice
Select the Number of Registered Deliveries
Generation of the Invoice Data
Registration of the Invoice in the System
Check of the Original Offer

Figure 2
Our tool's output (down) and the reference model (up) for low complexity of text

### 5.3.2 Medium Text Complexity

We chose the following example for testing with medium text difficulty:

***An offer is required to be done by the sales department of a specific company. At this point, the commercial manager has to perform a feasibility check based on the business requirements & needs to ensure that the offer is feasible. If the offer is not feasible, the customer has to be informed and at this point the process ends after informing him. If the offer is feasible, the personnel needs & availability has to be checked. After clarifying the availability based on the business resources, the offer has to be created by the commercial employee and the process ends at creating the offer.***

The BPMN model generated by our tool is shown in Figure 3 alongside the reference diagram from the course. The tool accurately identified all process steps, decision points, and branches, correctly assigning corresponding BPMN elements. The output captures the logical relationships and sequence described in the input text. The diagram is clear, intuitive, and closely resembles the author's layout. However, it lacks information about the decision conditions and branch usage.

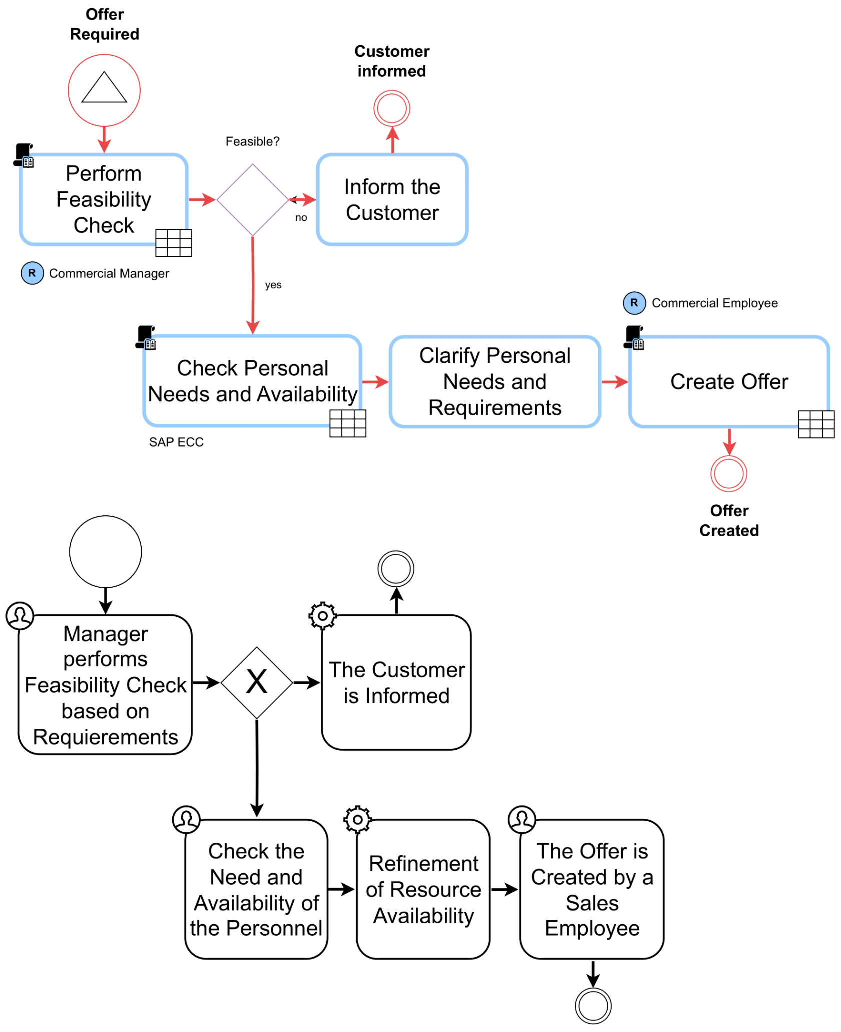
Offer Required
Customer informed
Feasible?
Perform Feasibility Check
no
Inform the Customer
Commercial Manager
yes
Commercial Employee
Check Personal Needs and Availability
Clarify Personal Needs and Requirements
Create Offer
SAP ECC
Offer Created
Manager performs Feasibility Check based on Requierements
The Customer is Informed
Check the Need and Availability of the Personnel
Refinement of Resource Availability
The Offer is Created by a Sales Employee

Figure 3
Our tool's output (down) and the reference model (up) for medium complexity of text

### 5.3.3 High Text Complexity

For testing with high text complexity, we chose the following example:

***Upon realizing that a book lending is required, the student has to request to lend the book. Afterwards, a reply on the book status will be sent and the book maybe either on loan/unavailable or available. If the book is available, the student has to submit the order to lend it and process ends with lending the book. On the other hand, if the book is unavailable/on loan, the student may either decide to***

***cancel the request if it is on loan for 1 month and the process ends with canceling the request or if the book is on loan for only 2 weeks the student will chose to hold the request and request to hold the book for the next reservation. After 2 weeks, the student will receive the availability of the book status and get the book. Finally, the process ends when the book is lent.***

The BPMN model generated by our tool is shown in Figure 4 along with the result of this task. While the tool identified all process steps and assigned the correct BPMN elements, it only partially met the test criteria. The output model failed to fully capture the logical relationships, with sequential flows not accurately reflecting the input text. Although the first decision point was correctly branched, subsequent branching was incorrect. The visual layout is unclear, and branching information is missing.

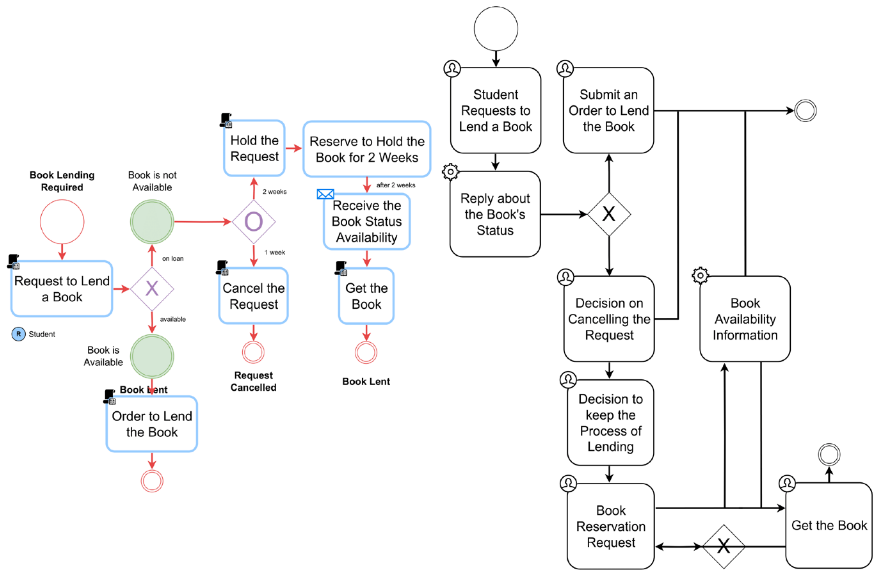

Figure 4
Our tool's output (right) and the reference model (left) for high complexity of text

## 6 Discussion

The obtained results confirm that the proposed LLM-based transformation pipeline provides a viable and accurate approach for converting unstructured textual process descriptions into formal BPMN models. This section discusses the significance of these findings, compares the proposed approach with existing techniques, and outlines the main limitations and directions for future work.

## 6.1 Comparison with Existing Approaches

Traditional methods for translating natural-language process descriptions into formal models typically rely on rule-based, pattern-matching, or ontology-driven extraction mechanisms. These approaches often depend on manually defined linguistic grammars or domain-specific templates to identify activity verbs, actors, and control-flow connectors. While such methods can achieve high precision in constrained contexts, they are notoriously brittle when confronted with linguistic variability, paraphrasing, or non-standard sentence structure.

In contrast, the proposed method employs a large language model (GPT) guided by structured prompting to infer the semantic relations implicit in textual descriptions. The key advantage lies in the model's ability to interpret diverse phrasing and resolve contextual dependencies without relying on manually curated rules. This allows for rapid adaptation to new process domains and text styles with minimal human intervention. The inclusion of an intermediate JSON layer, absent in most prior approaches, enables deterministic post-processing and validation, reducing the unpredictability of LLM outputs.

Compared to recent neural or hybrid systems that integrate syntactic dependency parsing with statistical learning, the proposed pipeline achieves comparable structural accuracy while maintaining full BPMN 2.0 compliance. Although fine-tuned domain-specific models may outperform general-purpose LLMs in niche applications, the presented results demonstrate that high-quality process models can be generated even with a general GPT model when guided by a well-engineered prompt and validation mechanism.

## 6.2 Limitations and Future Work

Despite its promising results, the current system has several limitations. First, the evaluation dataset comprises 30 process descriptions, which, although diverse, limits the statistical representativeness of the results. The categorization into simple, moderate, and complex cases offers useful insight, but larger-scale experiments are needed to confirm generalizability.

Second, the evaluation procedure relied on manual alignment between generated and reference diagrams, which ensures accuracy but restricts scalability. Developing automated comparison metrics that align BPMN graphs based on semantic similarity would enable broader benchmarking and reproducibility.

Third, handling of complex control-flow patterns remains a challenge. While the system performs well for simple and moderately complex processes, it struggles with nested gateways, loops, and event-driven subprocesses. Enhancing the prompt structure or integrating symbolic reasoning components could improve logical consistency in such cases.

Finally, domain-specific adaptation offers a promising avenue for future research. Fine-tuning or instruction-tuning the LLM on business process modeling corpora could significantly improve accuracy in complex scenarios. Integration with existing BPMN editing environments may also enhance usability for professional modelers.

## Conclusions

This paper presented a novel approach for transforming unstructured textual descriptions of business processes into formal BPMN 2.0 models using large language models. The proposed methodology combines guided prompting, structured JSON-based representation, and deterministic post-processing to ensure syntactic validity and semantic coherence of the resulting diagrams.

Evaluation on the WU Wien BPMN dataset confirmed the effectiveness of the approach, with mean F1 scores above 0.8 for simple and moderately complex process descriptions. The system consistently produced syntactically valid and structurally accurate BPMN models, demonstrating its potential as a practical assistant for semi-automated process modeling.

While performance decreases for complex narratives involving nested control-flow constructs, the overall transformation pipeline remains robust and interpretable. These results indicate that large language models, when combined with formal validation and layout automation, can bridge the gap between informal process documentation and formal model synthesis.

Future work will focus on extending the dataset, integrating automated evaluation metrics, and exploring domain-specific fine-tuning to improve accuracy in complex scenarios. The long-term goal is to integrate the proposed approach into collaborative BPMN tools. Where the human analysts can refine automatically generated models, thus creating a hybrid workflow that leverages both human insight and machine intelligence.

In conclusion, this study demonstrates that large language models can serve as reliable and efficient intermediaries between natural language and process modeling languages, opening new directions for research and practical deployment in intelligent business process design.

## Acknowledgement

This work was supported in part by the Slovak Research and Development Agency undercontract No. APVV SK-CZ-RD-21-0028 and APVV-21-0318, by Ministry of Education, Science, Research and Sport of the Slovak Republic, and the Slovak Academy of Sciences undergrant VEGA 1/0685/23 and 1/0353/26.

## References

[1] Mösslang M., Bernsteiner R., Ploder C., and Schlögl S.: Automatic Generation of a Business Process Model Diagram Based on Natural

Language Processing, Proceedings of the International Conference on Knowledge Management in Organizations, 2024, pp. 237-247

[2] Bilanová Z., Perháč J., Chovancová E., and Chovanec M.: Logic analysis of natural language based on predicate linear logic, Acta Polytechnica Hungarica, 2020, Vol. 17, No. 1, pp. 239-252

[3] Sholiq S., Sarno R., and Astuti E. S.: Generating BPMN diagram from textual requirements, Journal of King Saud University - Computer and Information Sciences, 2022, Vol. 34, No. 10, pp. 10079-10093

[4] Babaalla Z., Abdelmalek H., Jakimi A., and Oualla M.: Extraction of UML class diagrams using deep learning: Comparative study and critical analysis, Proceedings of Procedia Computer Science, 2024, Vol. 236, pp. 452-459

[5] Indurkhya N. and Damerau F. J.: Handbook of Natural Language Processing, Second Edition, Chapman & Hall/CRC, 2010

[6] Opitz J., Wein S., and Schneider N.: Natural Language Processing RELIES on Linguistics, arXiv preprint arXiv:2405.05966, 2024

[7] Polak M. P., Modi S., Latosinska A., Zhang J., Wang C.-W., Wang S., Hazra A. D., and Morgan D.: Flexible, model-agnostic method for materials data extraction from text using general purpose language models, Digital Discovery, 2024, Vol. 3, No. 6, pp. 1221-1235

[8] Hirtreiter E., Schulze Balhorn L., and Schweidtmann A. M.: Toward automatic generation of control structures for process flow diagrams with large language models, AIChE Journal, 2024, Vol. 70, No. 1, p. e18259

[9] Bellan P., Dragoni M., Ghidini C., van der Aa H., and Ponzetto S. P.: Process Extraction from Text: Benchmarking the State of the Art and Paving the Way for Future Challenges, arXiv preprint arXiv:2110.03754, 2021

[10] Qin L., Chen Q., Feng X., Wu Y., Zhang Y., Li Y., Li M., Che W., and Yu P. S.: Large language models meet NLP: A survey, arXiv preprint arXiv:2405.12819, 2024

[11] Licardo J. P., Tanković N., and Etinger D.: A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing, Proceedings of Procedia Computer Science, 2024, Vol. 239, pp. 483-490

[12] Jiang P., Rayan J., Dow S. P., and Xia H.: Graphologue: Exploring large language model responses with interactive diagrams, Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023, pp. 1-20

[13] Zhang T., Sahinidis N., Siirola V., and Jeffrey J.: Pattern Recognition in Chemical Process Flowsheets, AIChE Journal, 2019, Vol. 65, pp. 592-603

[14] Nousias N., Tsakalidis G., and Vergidis K.: Not yet another BPM lifecycle: A synthesis of existing approaches using BPMN, Information and Software Technology, 2024, Vol. 171, p. 107471

[15] Leopold, H., Mendling, J., & Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Transactions on Software Engineering, 2015, Vol. 41, No. 2, pp. 188-201

[16] Ambrosini, L., Soffer, P., & Bordbar, B.: Extracting control-flow from natural language texts: An evaluation of deep learning models. Information Systems, 2022, Vol. 107, p. 102025

[17] Sylvester, A., Reijers, H. A., & van der Aa, H.: Using large language models for process extraction: Capabilities and limitations. Decision Support Systems, 2024, Vol. 170, p. 113950

[18] Berti, A., van Zelst, S., & van der Aalst, W. M. P.: Automated process discovery using language models and constraint-aware refinement. ACM Transactions on Management Information Systems, 2023, Vol. 14, No. 3, pp. 1-27

[19] WU (Vienna University of Economics and Business), *BPMN Webtrainer Dataset – Exercises and Solutions*, ERP Systems Group, Vienna, Austria. Available online: https://www.wu.ac.at/erp/education/webtrainer/bpmn (Accessed: Nov. 10, 2025)