


Modeling meets Large Language Models

Martin Forell ¹ and Selina Schüler ¹

Abstract:



Modeling business processes is often challenging due to its complexity and potential for errors. One key issue arises when process experts and modelers are different individuals, which can lead to communication gaps and result in low-quality business process models. Recognizing this, our paper prioritizes the initial phase of modeling in Business Process Management (BPM). We propose a method that leverages Large Language Models (LLMs) to efficiently transform written business process descriptions into comprehensive graphical models. This approach offers a standardized and streamlined procedure to enhance the quality and effectiveness of business process modeling. While we focus on Petri nets as a primary example, our approach is adaptable to other graphical modeling languages. We present a novel method involving a series of LLMs to extract essential data, setting the stage for creating various graphical models. This technique aims to generate initial drafts that can be further refined, and its sequential application allows for adaptability to different modeling tools, including but not limited to the Horus Business Modeler.

Keywords: Large Language Model, Automatic Business Process Model Generation, Petri nets

1 Introduction

The rapid advancements in Natural Language Processing (NLP), especially in the domain of Large Language Models (LLMs) have opened new opportunities in various fields, including Business Process Management (BPM). Initial experiments, like those documented in [FFK23], have showcased the capabilities of tools such as OpenAI's ChatGPT [Op22] in creating specific process models from given prompts. This paper delves into the deployment of LLMs across the business process lifecycle, emphasizing their dynamic nature and adaptability.

The structure of this paper is as follows: Sect. 2 motivates the use of LLMs in BPM. Sect. 3 provides an introduction to NLP, focusing on the concept of prompt engineering and LLMs. In Sect. 4, we present the methodological framework adopted in this paper for incorporating LLMs into business process modeling, including a detailed discussion of the implementation and evaluation of this approach. Finally, Sect. 5 concludes the paper, offering insights into future research directions and potential improvements to our implementation.

¹ Karlsruhe Institute of Technology (KIT), Institute of Applied Informatics and Formal Description Methods (AIFB), Kaiserstr. 89, 76133 Karlsruhe, Germany,
martin.forell@kit.edu,  <https://orcid.org/0000-0003-4512-6144>;
selina.schueler@kit.edu,  <https://orcid.org/0000-0003-3498-9382>
This work is licensed under Creative Commons Attribution 4.0 International License <http://creativecommons.org/licenses/by/4.0/>, <https://doi.org/10.18420/modellierung2024-ws-003>

2 Business Process Management including Large Language Models

The business process lifecycle, as illustrated in Fig. 1, consists of phases that are interrelated. The phases are arranged in a cyclical configuration, effectively highlighting their logical interdependencies. These dependencies do not imply a strict temporal order in which the phases need to be executed [We19].

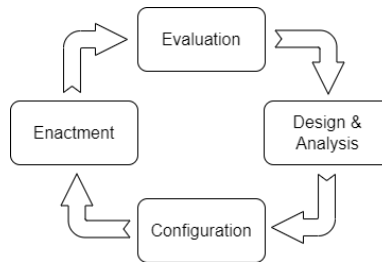


Fig. 1: Business process lifecycle [We19]

Various approaches have been proposed in research and practice to partially or fully automate the creation of business process models. [SA24] conducted a systematic literature review (SLR) on automatic generation of business process models or [De18] and [Ma19] specially for NLP in business process modeling. The SLRs demonstrate that there are different approaches for the automatic generation of business process models, which differ in terms of input data, generation methods and modeling languages used. Despite active research in the field of automatic model generation, there is still considerable potential for improvement. New technologies are constantly being developed to support automatic model generation. LLMs offer a solution whereby users can input data like textual process descriptions, transforming them into process models [FFK23]. Prospects for the future include extending this capability to handle a broader range of data beyond mere process descriptions. Therefore, LLMs provide valuable support by establishing a solid starting point for further process exploration based on user-provided data. Once the design phase is completed, the business process model can be analyzed and improved. Various approaches have been developed to enhance the quality of these models by evaluating their syntactic, semantic, and pragmatic aspects, as discussed in [Co18]. In addition, alternative models are created based on the model to find more efficient ways to accomplish certain tasks [TM19]. LLMs could facilitate this by allowing users to ask questions in natural language, to which a possible answer is calculated based on the available data. The LLM then provides the most probable response, even in instances where it may not precisely align with the posed question. By uploading the initial process model along with other reference models or guidelines for effective modeling, users can obtain suggestions for model adaptation. For instance, if activities are independent, suggesting concurrency becomes possible.

In order to include all phases in the analysis for the inclusion of LLMs in BPM, the business process must be implemented in the configuration phase after the business process model has been designed and analyzed [We19]. In this phase, to facilitate the integration of Artificial Intelligence (AI), there are methodologies available for explicitly

modeling business processes that incorporate AI. [Ta24] present a comprehensive tool concept for the step-by-step support of explicit modeling the integration of Machine Learning (ML) applications in business processes modeled with Business Process Model and Notation (BPMN), so that actual process models without ML can easily be extended to process models with ML. More formal modeling of ML cases allows more options for analyzing and using (e.g., executing) models. If the process becomes operational, LLMs can continue to support process participants by answering questions based on the process model or additional provided data. For instance, users might inquire about the consequences of an incorrect invoice, and the LLM provides the most probable answer, which can be annotated in the model. The implementation phase of BPM is followed by the evaluation phase. LLMs can contribute to process improvement by using real process data in a similar way to the introductory analysis stage. However, their effectiveness depends on the quality of the underlying data.

3 Natural Language Processing

Natural Language Processing (NLP) is a subfield of AI and ML that focuses on enabling computers to understand, interpret and generate human language, using techniques from computer science and computational linguistics [Kh23; Zh23]. This domain has experienced substantial growth over the past few years, largely driven by progress in ML [Ba22; Ka20; Op23; Ou22; Va17]. According to [Li21], applications in NLP can be categorized into five types, each encompassing various tasks such as sentiment analysis, natural language inference, named entity recognition, or text summarization. However, for this paper, only the task of text generation is relevant.

3.1 Large Language Models

Advancements in NLP have led to the emergence of LLMs as cutting-edge AI systems, renowned for their coherent text processing and generalization capabilities across multiple tasks [Ar22; Ra18]. These sophisticated generative models statistically represent the distribution of tokens, including whole words, word fragments, individual characters, and punctuation marks, in extensive human-generated text corpora [Sh23]. Central to the effectiveness of these LLMs is the Transformer architecture, an influential neural network design that has become a staple in modern NLP tasks [Va17; Zh23].

First introduced by Google Brain in 2017, the Transformer architecture [Va17] marked a significant milestone in NLP, particularly in translation tasks. Unlike its predecessors, the Transformer architecture offers a more structured memory system, better handling the long-term dependencies typical in natural text sequences [Va17]. A critical innovation within the Transformer architecture is the Multi-head Self-attention layer [Va17]. This layer enables the model to concurrently process information from various representation subspaces at different points in the architecture, a feature articulated by [Va17]. The architecture's design,

characterized by shorter internal signal paths, facilitates more effective learning of long-term dependencies in text. This, in turn, allows for a deeper understanding of the semantics of natural language, as further discussed by [Zh23]. Consequently, today's cutting-edge Transformer-based models, including GPT-4 [Op23] and ChatGPT [Op22], all stem from the foundational principles of the Transformer architecture, albeit with specific architectural enhancements. These models not only embody the original strengths of the Transformer design but also demonstrate its versatility and ongoing evolution in the rapidly advancing field of NLP.

3.2 Prompt Engineering

LLMs generate outputs through inputs in natural language, termed prompts, which essentially act as directives to elicit specific responses from LLMs [Wa23]. This technique, known as prompting, encapsulates the user's intent in a query, thereby guiding the LLM to produce the expected outcome by establishing specific parameters and instructions [Wa23]. An illustrative example of prompt engineering in action is shown in *Example: Prompt 1*. Based on [Wa23], this example demonstrates how the LLM engages in a focused dialogue, asking specific questions about a business process description. The LLM persists in this interactive process until it has collected enough information to effectively model a Petri net. This case highlights the LLM's ability to proactively seek out context-relevant information, showcasing the versatility and expanded capabilities of prompt-based programming. Moving beyond basic tasks like "generate a method that does X" or straightforward question-answering, this example underscores the sophisticated application possibilities of prompt engineering [Wa23].

Example: Prompt 1

```
Please review the business process description I have provided
and ask me clarifying questions to better understand it. Once you
have gathered sufficient information, identify and extract the key
components necessary for modeling this as a Petri net.
```

Upon extracting the necessary data for a specific task, like generating business process models, one methodology is OpenAI's function calling feature [AJL23]. These advancements enable the LLM to generate JSON objects specific to certain functions, enhancing the integration between the capabilities of LLMs such as *GPT-4* and various external tools or Application Programming Interfaces (APIs) [AJL23].

An example to demonstrate the function calling feature can be found in Listing 1, which presents the schema for a Petri net in JSON format. This schema defines the basic structure of data for a Petri net, focusing on essential components such as *places*, *transitions*, and *edges*. It simplifies the representation by treating places and transitions as arrays of strings, where each string represents a unique identifier for a place or transition. The *edges* are defined as objects, specifying connections with a *from* and a *to* property, both of which are strings. This structure captures the fundamental relationships within a Petri net without

delving into detailed attributes like marking conditions or weights. The schema ensures that any JSON data conforming to it must include these key elements.

```

1  {"$schema": "http://json-schema.org/draft-07/schema#",
2   "type": "object",
3   "properties": {
4     "places": {
5       "type": "array",
6       "items": {"type": "string"}},
7     "transitions": {
8       "type": "array",
9       "items": {"type": "string"}},
10    "edges": {
11      "type": "array",
12      "items": {
13        "type": "object",
14        "properties": {
15          "from": {"type": "string"},
16          "to": {"type": "string"}},
17        "required": ["from", "to"]}},
18    "required": ["places", "transitions", "edges"]}

```

List. 1: JSON Schema Example Illustrating the Structure for a Petri net.

This JSON Schema, when combined with a LLM like OpenAI's *GPT-4-0125-preview* and provided with a prompt, such as shown in *Example: Prompt 2*, can be utilized to generate structured output in the form of JSON, demonstrating the practical application of this schema.

Example: Prompt 2

A Petri net modeling a simple decision process with two possible outcomes and a single decision point.

Listing 2 exemplifies a JSON instance that conforms to the previously defined Petri net schema and is generated by using the mentioned prompt. It illustrates how a simple decision process within a Petri net is represented in JSON, adhering to the schema's structural constraints. This example serves as a clear demonstration of the LLM's capability to interpret prompts and generate structured, schema-compliant data.

```

1  {"places": [
2   "Start",
3   "Outcome1",
4   "Outcome2"],
5   "transitions": [
6   "DecisionPoint"],
7   "edges": [
8   {"from": "Start", "to": "DecisionPoint"},
9   {"from": "DecisionPoint", "to": "Outcome1"},
10  {"from": "DecisionPoint", "to": "Outcome2"}]}

```

List. 2: JSON Representation of a Petri net. Generated by using OpenAI's Functions Feature.

4 Framework, Implementation and Evaluation

This section outlines our methodological framework, details the implementation of our prototype, and presents a thorough evaluation of its performance. We introduce an approach leveraging LLMs for business process modeling, focusing on the strategic use of prompt engineering and function calling.

4.1 Framework empowering Large Language Models for business process modeling

In this paper, we introduce a business process model generation approach using LLMs, prompt engineering and the utilization of function calling capabilities. The approach involves utilizing instructional prompts, which clearly define the main task for the LLM for the extraction of relevant business process information (1), provide explicit guidelines for specific modeling languages to generate a business process model based on the extracted information (2) and define a format that is compatible with modeling tools to transform the model into a specific export format (3). Lastly, the generated output is checked against defined model properties (4). These steps are outlined in Fig. 2.

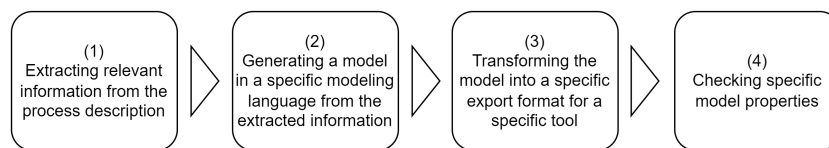


Fig. 2: Business process model generation approach using LLM.

As we focus on generating models from business process descriptions, the LLM might be instructed to actively query the document provided. This procedure grants the LLM a preliminary understanding of the business process in question. A business process is a set of manual, semi-automated or automated activities that are executed in a company according to specific rules, towards a specific goal [Ob96]. In particular, business process models depict the temporal-logical sequence of activities [Fo10]. The temporal-logical sequential conditions of activities result from the relationships between the activities within a process. These relationships can be causal conditions or organizational specifications. Two activities that are causally related must take place in a certain order. Causal relationships between the activities of a process can be given, for example, by the flow of documents, forms, files (folders) or process folders. Activities can also be mutually exclusive within a process - e.g. if they use the same resource. Finally, activities can also take place causally independently of each other ("concurrently") [Ob96]. In order to generate a business process model, the information on the activities and their interrelationships must be extracted. Therefore, the prompt should include that the following information is explicitly requested in the process description: States of the process, activities of the process, relationships between the activities such as sequences, cycles, concurrency and alternatives.

The next step is to transform this information into a model in a specific modeling language. Therefore, the prompt should describe the selected modeling language. This includes the definition of the modeling language and the specification of how the extracted information is mapped in a model in the respective modeling language. As we are focusing on Petri nets in this work, these are defined as a bipartite graph consisting of places, transitions and edges. In addition, the prompt must define how sequences, cycles, concurrency and alternatives are mapped in a Petri net. For example, concurrent paths are represented by splits at transitions and alternative paths by splits at places.

An external tool can be used to display a model graphically. Therefore, in this approach, the generated model is transformed into a defined exchange format. To do this, it must be specified in the prompt how the identified nodes and edges of the model are described in the desired exchange format.

In order to improve the model quality, properties can also be formulated in the prompt that should be checked and adapted if necessary. These include, for example, that source and sink node of the Petri net should be places (where the marking with a token indicates start and end state of the process).

4.2 Implementation for Petri nets

Our methodology for enhancing LLMs in business process modeling is realized through a two-phase software prototype, as illustrated in Figure 3. The initial phase, named Pre-Processing, encompasses the analysis and organization of the business process description. Subsequently, the prototype transitions to the Processing phase, wherein the Petri net is constructed. The full set of prompts is available at <https://github.com/KIT-BIS/bpm-tool>.

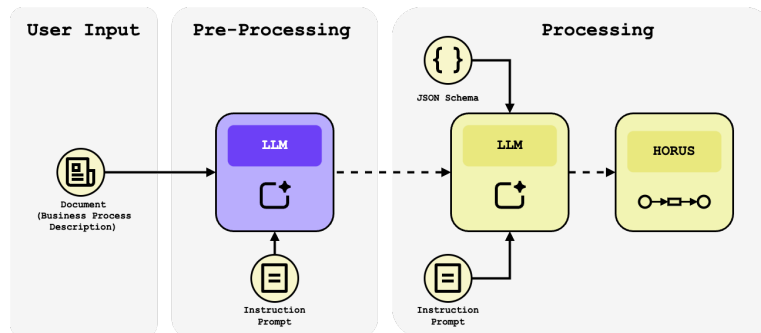


Fig. 3: Overview of the Prototype's Architecture.

Phase 1: In the initial Pre-Processing phase an instruction prompt is provided to the LLM. This phase plays a crucial role in the interpretation and organization of the input. As illustrated in Fig. 4, the instruction prompt consists of four distinct parts and includes the description of the business process as input for the analysis. This clear direction ensures that the model understands its intended purpose. First, the objective is delineated, providing a

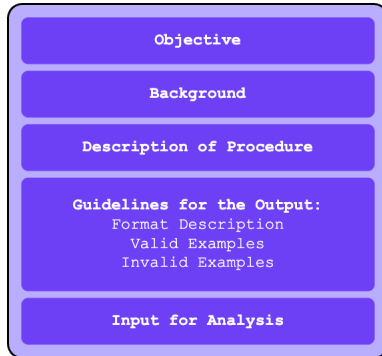


Fig. 4: Sections of the Instruction Prompt: Pre-Processing

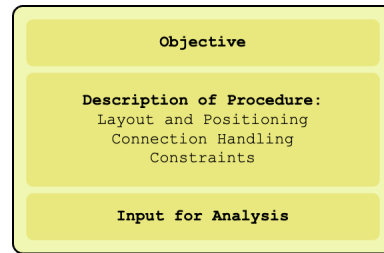


Fig. 5: Sections of the Instruction Prompt: Processing

foundational understanding necessary for the task. This objective directs the attention of the LLM toward its designated task and establishes a baseline understanding. Subsequently, a comprehensive and clear definition of a Petri net is presented as background information for the LLM. This should encompass an explanation of our understanding of a Petri net. After defining the objective and background, the description of procedure outlines the procedural steps involved in the analysis of the given input. It includes concurrency detection, title determination, place and transition identification, edge analysis, examination of network characteristics, and adherence to additional constraints. Each step is described in detail, emphasizing the importance of recognizing patterns, naming elements, and ensuring strong connectivity of the Petri net. The final segment of the instruction prompt specifies the output format and provides some valid and invalid examples to the LLM. It details how concurrency detections, places, transitions, and edges should be documented and formatted.

Phase 2: In the Processing phase, the LLM interprets the output from the Pre-Processing phase. Its goal is to transform this output into a format compatible with various modeling tools. By adapting the prescribed output JSON schema, our approach can be flexible and used for different tools. We demonstrate its application using the Horus Business Modeler [Ho23] and the JSON-Nets Editor [FFS23], with a focus on Petri net visualization. An established JSON Schema, detailing the required input format for the modeling tool, is combined with the output from Phase 1 and supplemented by an instructional prompt. This prompt, as shown in Fig. 5, outlines the transformation objectives and the steps necessary for constructing the Petri net. It also includes guidance on visualizing the Petri net, instructing the LLM on the placement of elements to ensure clarity and coherence in the final model. Our implementation accommodates the use of various LLMs, such as OpenAI's *GPT-3.5-turbo*, *GPT-3.5-turbo-1106*, *GPT-4*, and *GPT-4-0125-preview*. Additionally, we tested different open-source LLMs such as *Mistral 7B Instruct* [AI23a] and *Mixtral 8x7B Instruct* [AI23b]. For the purposes of this paper, we utilized *GPT-4* and *GPT-4-0125-preview*, as they demonstrated the best performance in our tasks. The selection of these models was

based on their enhanced capabilities in accurately interpreting and processing business process information, crucial for the effective construction and visualization of Petri nets.

4.3 Evaluation

In this section, we assess the efficacy and accuracy of our implementation in generating Petri nets from textual descriptions of business processes. The aim is to critically analyze the system's ability to accurately interpret the process information, manage concurrency and decision points, and translate these into coherent Petri nets. We focus on the syntactic correctness (ensure it adheres to the defined syntax rules and standards), semantic correctness (accurate identification of process elements and relationships) and pragmatic correctness (usability and visual coherence of the generated models) of the outputs.

Evaluation strategy

Our evaluation strategy involves testing the system with a series of examples, each representing a different complexity level and structure of the business process description.

Process Description: Example 1

<p>The process begins with a customer visiting the online bookstore, place is called 'Bookstore visited'. The customer places an order for a book. The order now is in the 'Order Received' place. The bookstore staff receives the order. The 'Process Order' transition is triggered, where they verify the order details, locate the book in their inventory, package it, and prepare it for dispatch. The process moves the order from 'Order Received' to an intermediate state indicating that the order is being processed. Once the order is ready for dispatch, the 'Dispatch Order' transition is activated. This transition represents the act of handing the package over to the delivery service. The order status now moves to the 'Order Dispatched' place. The customer is notified that the order has been dispatched, completing the process.</p>

Example 1 should represent a sequential process with clearly defined stages, making it an ideal case to test the basic capabilities of our system to capture sequential process flows. The evaluation focuses on the system's ability to correctly identify and sequence the stages, as well as the transitions between these stages. Therefore, *Example 1* represents a standard online bookstore process, beginning with a customer visit and ending with the dispatch of an order.

Process Description: Example 2

A customer visits the online bookstore and places an order for a book. The bookstore staff receives the order. The order is being processed and the bookstore staff verifies the order details, locates the book in their inventory, packages it, and prepares it for dispatch. Once the order is ready for dispatch, the package is handed over to the delivery service. The customer is notified that their order has been dispatched, completing the process.

Example 2 presents a condensed version of the bookstore process of *Example 1*. It challenges the system's ability to handle a more compressed narrative and infer the underlying process structure. The key aspects of evaluation here are the system's proficiency in extrapolating the necessary stages from a less detailed description and its effectiveness in maintaining the logical flow of the process.

Process Description: Example 3

A customer places an order for a book on the online bookstore. The order is received. During the processing of the order, the order details are verified and the stock availability is checked. If the book is in stock, the order is processed. If not, the customer is informed about the delay and options for backorder or substitution.

Example 3 is intended to introduce complexity in the form of conditional processing by providing alternative paths in the process description. It tests the system's capability to handle branching scenarios and decision-making processes. The evaluation focuses on how well the system identifies and represents these decision points, particularly the handling of the 'in stock' versus 'out of stock' scenarios, and their impact on the overall process flow in the Petri net.

Evaluation results

Fig. 6a displays the Petri net generated from the online bookstore process described in *Example 1*. The implementation effectively identified and sequenced key stages like *Bookstore Visited*, *Order Received*, *Order Processed*, and *Order Dispatched*, along with their corresponding transitions. The Petri net correctly reflects the process's sequential flow, indicating a strong semantic understanding of the described procedure. Syntactically, the net adheres to standard Petri net conventions, with logical connections and a coherent structure, making it easy to interpret.

Fig. 6b illustrates the Petri net generated from the process description in *Example 2*. The system generated a syntactically correct Petri net. The net adheres to the conventional structure of Petri nets with places and transitions logically connected. This outcome confirms the system's capability to structure process information into a Petri net, even when the input is less explicit. Notably, the model generated is more complex compared to the first example. This complexity stems from the system's attempt to infer process stages and transitions from a narrative that lacks explicit details. This observation underscores the importance of detailed process descriptions for generating simpler and more accurate Petri nets. Despite

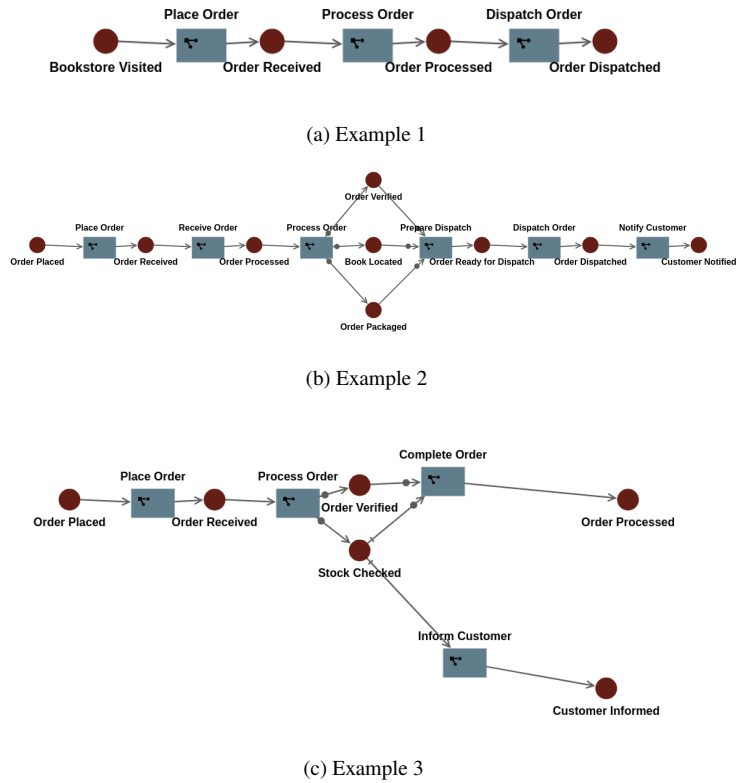


Fig. 6: Generated Petri nets for the given examples.

its syntactical correctness, the model reveals areas needing semantic improvement. A prominent instance is the premature introduction of a place labeled *Order placed* before the transition *place order*, indicating a misinterpretation of the process sequence. Additionally the transition *Process Order* could be represented by three individual transitions (verify order; locate book, pack order). This issue points to a limitation in the system’s ability to accurately infer and sequence events from a brief narrative.

Fig. 6c illustrates the Petri net generated from the process described in *Example 3*. The net adheres to the conventional structure of Petri nets, with places and transitions logically connected. The model managed to identify a AND-Split, but did not generate it completely correct. Despite its syntactical correctness, the model reveals areas needing semantic improvement. The complete collection of images pertaining to Examples 1, 2, and 3, which have been generated based on the JSON-Nets Editor’s JSON-Schema and imported into the JSON-Nets Editor, can be accessed at the following URL: <https://github.com/KIT-BIS/bpm-tool>.

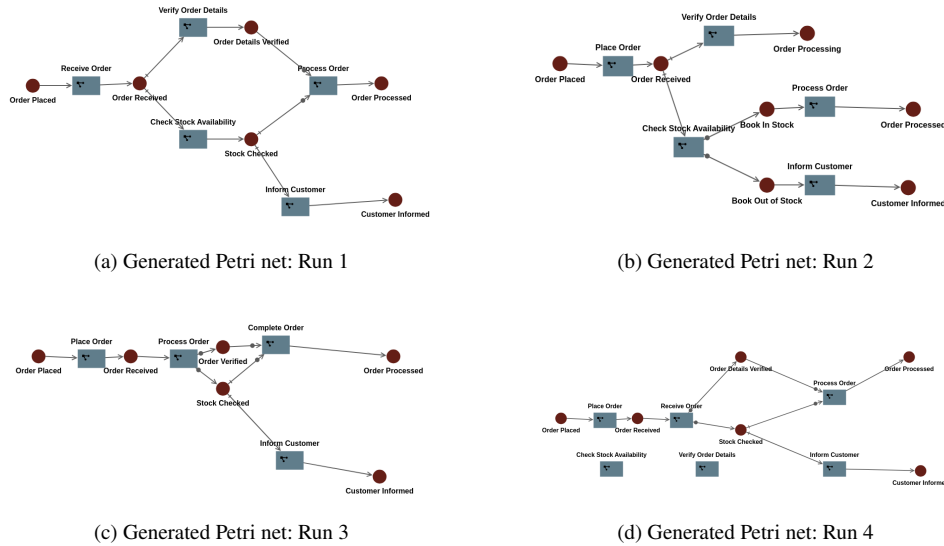


Fig. 7: Overview of different generated Petri nets for different runs for *Process Description: Example 3*

The evaluation of the three examples provides valuable insights into the capabilities and limitations of our system in generating Petri nets from textual descriptions of business processes. Across the examples, the system demonstrated the ability to produce Petri nets that are syntactically correct. If there is a clear process description, the net is also semantically correct. The naming conventions that transitions that represent activities are actively named (e.g. place order) and places that represent states are passively named (e.g. order placed) were adhered to. This affirms the system’s proficiency in accurately interpreting and structuring process information into coherent Petri net models. However, the evaluation also highlighted that achieving deterministic output continues to be a challenge with LLMs. This characteristic was evident in the system’s tendency to produce different solutions upon each execution. Fig. 7 illustrates this, showcasing four varied solutions from four separate runs. Such variability, while indicative of the flexible and adaptive nature of LLMs, also poses challenges in ensuring consistent output, which is crucial for reliable process modeling. The multiple runs also show that it is particularly difficult to model concurrency or alternatives and to merge them semantically correctly.

5 Outlook

In this paper, we introduced a business process model generation approach using LLMs, prompt engineering and the utilization of function calling capabilities. Firstly, the approach involves the use of instructional prompts, which clearly define the main task for the LLM. Secondly, the approach entails providing clear directives tailored to specific modeling

languages. Thirdly, the approach includes defining a format that is seamlessly compatible with modeling tools. Lastly, the approach includes defining model properties which should be checked. The procedural steps of this approach are shown in Figure 2.

We acknowledge the challenges faced by the current system, particularly when dealing with intricate process structures that involve branching and parallelism. The system's limitations were most evident when it came to generating accurate representations of XOR-Splits/XOR-Joins and AND-Splits/AND-Joins, with occasional misinterpretations leading to potential inaccuracies in the resulting Petri nets.

In the future, we aim to implement several enhancements to our method and software tool, particularly focusing on refining the conversion of textual business process descriptions into Petri nets. Our primary objectives will be to develop sophisticated techniques for managing complex structures and to improve the consistency of the outputs. These improvements are crucial for boosting the overall performance and usefulness of the system.

Moreover, future iterations should expand the scope of testing to encompass additional modeling languages, such as BPMN and extending compatibility to other modeling tools beyond the Horus Business Modeler. Developing a specifically fine-tuned model could improve the recognition of specific modeling languages, thereby enhancing the quality of outputs. Incorporating human feedback could also significantly refine the information extraction process. By enabling a collaborative interaction where either the LLM or the user can pose clarifying questions or request refinements, we can move toward a more robust and interactive system that leverages the strengths of both AI and human intelligence for better business process modeling.

References

- [AI23a] AI, M.: Mistral 7B, en-us, Section: news, 2023, URL: <https://mistral.ai/news/announcing-mistral-7b/>, visited on: 02/23/2024.
- [AI23b] AI, M.: Mixtral of experts, en-us, Section: news, 2023, URL: <https://mistral.ai/news/mixtral-of-experts/>, visited on: 02/23/2024.
- [AJL23] Atty Eleti; Jeff Harris; Logan Kilpatrick: Function calling and other API updates, en-US, 2023, URL: <https://openai.com/blog/function-calling-and-other-api-updates>, visited on: 01/07/2024.
- [Ar22] y Arcas, B. A.: Do Large Language Models Understand Us? *Daedalus* 151/2, pp. 183–197, 2022.
- [Ba22] Black, S.; et al.: GPT-NeoX-20B: An Open-Source Autoregressive Language Model. In: *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*. Association for Computational Linguistics, virtual+Dublin, pp. 95–136, 2022.
- [Co18] Corradini, F.; Ferrari, A.; Fornari, F.; Gnesi, S.; Polini, A.; Re, B.; Spagnolo, G. O.: A Guidelines framework for understandable BPMN models. In: *Data & Knowledge Engineering*. Vol. 113, pp. 129–154, 2018.

- [De18] De Almeida Bordignon, A. C.; Thom, L. H.; Silva, T. S.; Dani, V. S.; Fantinato, M.; Ferreira, R. C. B.: Natural Language Processing in Business Process Identification and Modeling: A Systematic Literature Review. In: Proceedings of the XIV Brazilian Symposium on Information Systems. ACM, Caxias do Sul Brazil, pp. 1–8, 2018.
- [FFK23] Fill, H.-G.; Fettke, P.; Köpke, J.: Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. In: Enterprise Modelling and Information Systems Architectures (EMISAJ). Vol. 18, pp. 1–15, 2023.
- [FFS23] Fritsch, A.; Forell, M.; Schüler: KIT-BIS/json-nets, original-date: 2023-01-13T15:15:30Z, 2023, URL: <https://github.com/KIT-BIS/json-nets>, visited on: 02/22/2024.
- [Fo10] Foth, E.: Exzellente Geschäftsprozesse mit SAP: Praxis des Einsatzes in Unternehmensgruppen. Springer-Verlag, Berlin, Heidelberg, 2010.
- [Ho23] Horus software GmbH: Business Modeler, 2023, URL: <https://www.horus.biz/de/produkte/business-modeler/>, visited on: 01/08/2024.
- [Ka20] Kaplan, J.; et al.: Scaling Laws for Neural Language Models, 2020, URL: <https://arxiv.org/abs/2001.08361>, visited on: 01/05/2024.
- [Kh23] Khurana, D.; Koli, A.; Khatter, K.; Singh, S.: Natural language processing: state of the art, current trends and challenges. In: Multimedia Tools and Applications. Vol. 82. 3, pp. 3713–3744, 2023.
- [Li21] Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; Neubig, G.: Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, 2021, visited on: 12/08/2023.
- [Ma19] Maqbool, B.; et al.: A Comprehensive Investigation of BPMN Models Generation from Textual Requirements—Techniques, Tools and Trends. In (Kim, K. J.; Baek, N., eds.): Information Science and Applications 2018. Vol. 514, Springer Singapore, Singapore, pp. 543–557, 2019.
- [Ob96] Oberweis, A.: Modellierung und Ausführung von Workflows mit Petri-Netzen. Vieweg+Teubner Verlag, Wiesbaden, 1996.
- [Op22] OpenAI.: Introducing ChatGPT, en-US, 2022, URL: <https://openai.com/blog/chatgpt#OpenAI>, visited on: 12/07/2023.
- [Op23] OpenAI et al.: GPT-4 Technical Report, 2023, URL: <http://arxiv.org/abs/2303.08774>, visited on: 01/05/2024.
- [Ou22] Ouyang, L.: Training language models to follow instructions with human feedback. In: Advances in Neural Information Processing Systems. Vol. 35, pp. 27730–27744, 2022.

- [Ra18] Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.: Language Models are Unsupervised Multitask Learners, 2018, URL: <https://techbooky.com/wp-content/uploads/2019/02/Better-Language-Models-and-Their-Implications.pdf>, visited on: 01/07/2024.
- [SA24] Schüler, S.; Alpers, S.: State of the Art: Automatic Generation of Business Process Models. In (De Weerd, J.; Pufahl, L., eds.): Business Process Management Workshops. Vol. 492, Springer Nature Switzerland, Cham, pp. 161–173, 2024.
- [Sh23] Shanahan, M.: Talking About Large Language Models, 2023, URL: <https://arxiv.org/abs/2212.03551>, visited on: 01/05/2024.
- [Ta24] Take, M.; Becker, C.; Alpers, S.; Oberweis, A.: Modeling the Integration of Machine Learning into Business Processes with BPMN. In (Yang, X.-S.; Sherratt, R. S.; Dey, N.; Joshi, A., eds.): Proceedings of Eighth International Congress on Information and Communication Technology. Vol. 696, pp. 943–957, 2024.
- [TM19] Tikhonov, S. E.; Mitsyuk, A. A.: A Method to Improve Workflow Net Decomposition for Process Model Repair. In (Van Der Aalst, W. M. P.; et al., eds.): Analysis of Images, Social Networks and Texts. Vol. 11832, pp. 411–423, 2019.
- [Va17] Vaswani, A.; et al., I.: Attention Is All You Need. In: Neural Information Processing Systems. Vol. 30, Curran Associates Inc, Red Hook, NY, 2017.
- [Wa23] White, J.; et al.: A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT, 2023, URL: <https://arxiv.org/abs/2302.11382>, visited on: 12/08/2023.
- [We19] Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer Berlin Heidelberg, Berlin, Heidelberg, 2019.
- [Zh23] Zhang, A.; Lipton, Z. C.; Li, M.; Smola, A. J.: Dive into Deep Learning. Cambridge University Press, 2023.