The paper has been published at the Wirtschaftsinformatik 2024:
https://wi2024.de/wissenschaftliches-programm/process-science/

How Can Generative AI Empower Domain Experts in Creating Process Models?

Nataliia Klievtsova[1], Juergen Mangler[1], Timotheus Kampik[2], Janik-Vasily Benzin[1], and Stefanie Rinderle-Ma[1]

[1] Technical University of Munich, TUM School of Computation, Information and Technology, Garching, Germany, {firstname.lastname}@tum.de
[2] SAP Signavio, Berlin, Germany, {firstname.lastname}@sap.com

Abstract. Considering the human factor in information systems is a key to future digitalization efforts, as stated in the Industry5.0 research and innovation actions of the EU. Especially in the design phase of a process-oriented information system, the human factor includes the empowerment of domain experts in process model creation lowering the entry hurdle for process modeling, and increasing modeling speed. In this work, we investigate how generative AI methods can support domain experts in creating process models in interaction with a chatbot based on textual process descriptions. We explore the amount of necessary information required as input to create process models with immediate visual representation using markdown-inspired languages and extend existing evaluation methods for assessing generated models, focusing on their completeness and correctness. Overall, an evaluation method has to consider the complex relationships between model completeness, correctness, textual process description, textual representation, and prompt engineering to support the domain expert.

Keywords: Digital Transformation · Business Process Engineering and Management · Process Modeling · Generative AI

1 Introduction

The design of process documentation and the creation of corresponding process models can be a complex and time-consuming task [6]. Yet, in conditions where an organization is an "open" system, constantly undergoing adaptations caused by internal or external changes, business processes are continuously redesigned based on their performance in day-to-day execution [14], requiring constant (re-)documentation and modeling.

One of the challenges during process modeling is the acquisition of knowledge from domain experts, considering that knowledge is often tacit and that even explicit knowledge is frequently distributed across a variety of sources and representation formats [41]. Domain experts might prefer process models in natural language (i.e., process descriptions), especially if they have not enough experience in and knowledge of (graphical) process modeling [5,32]. Nevertheless, there are several reasons why utilizing process models can still be beneficial for domain experts and other stakeholders (e.g., documenting, analyzing, and improving business processes). Process models can help domain experts and other stakeholders facilitate a clearer understanding of the underlying process, particularly if its complexity is high. In addition, process models can bring more clarity: although on surface level, natural language in a *lingua franca* such as English may be almost universally understood in a given organization, the ambiguity of natural language can lead to misunderstandings.

The traditional approach to process modeling, where only the process modeler carries out the model design based on her interpretation of the domain experts' views, bears several disadvantages. First, the modeling expert can become a filter and bottleneck in capturing the process knowledge from the stakeholders [29], leading to wrong modeling decisions. Additionally, the absence of a direct involvement of the domain expert might have a negative impact on its quality. Moreover, the emergence of psychological ownership during the domain expert's participation in the modeling process could have a positive effect on affective commitment [26].

Consequentially, domain experts might be reluctant to input information into the first version of a model due to lack of modeling skills [17]. Hence, the ability to convert natural language text into business process models holds considerable appeal, offering added value to the organization, i.e., by overcoming the communication gap not only between process modelers and domain experts during the process model generation phase [33], but also among diverse stakeholders involved in the actual process. As first studies show [24], Large Language Models (LLM) may offer a simple and effective way to automatically generate process models from text. **The overarching question of this work is thus how and to which degree LLMs can generate process models from process descriptions with sufficient process model quality.**

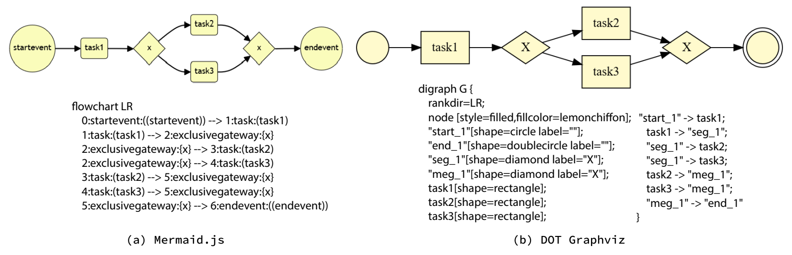
This study focuses on the evaluation of the quality of process models resulting from an LLM-based generation based on provided process descriptions, and is a continuation of our first analysis provided in [30] which focused on task extraction. To this end, we provide a method to generate BPMN 2.0 (`bpmn.org`) process models from textual process descriptions based on different LLMs (cf. Sect. 2). Although, at first glance, it seems that GPT is capable of creating some process models in `.bpmn` format (XML-based), this output does not contain crucial layout information, and the output becomes too long even for simple process models with fewer tasks. In addition, to our knowledge, there are no evaluation studies exploring the quality of these generated models. Hence, we opt for a combination of LLM and graphical tools such as Mermaid.js and Graphviz, reflected in the prompts. The quality of the generated process models is assessed based on completeness (quantitative) and correctness (qualitative) (cf. Sect. 3), yielding, in turn, a way to assess different LLMs w.r.t. their ability to generate process models. The evaluation is based on two detailed data sets containing both textual process description and associated graphical process models created by humans. Related work is discussed in Sect. 5, before Sect. 6 concludes this work.

2 How to teach BPMN 2.0 to LLMs?

This section provides the building blocks for control flow extraction by LLMs from text, i.e., the selection of suitable graphical representations and prompt engineering.

Selection of Graphical Representations: Due to the context length limitation of LLMs, it is not possible to utilize standard XML serialization of BPMN models: a regular language model input/output limit is between $1,000$–$8,000$ tokens. At the same time, even a simple BPMN model with 7 tasks in XML representation might exceed $10,000$–$15,000$ tokens [3], depending on its complexity. Therefore, it is required to work with an abstract graphical representation of the BPMN model. Multiple comprehensible graphical representations (GR) of various types of event logs or process models (e.g., directly-follows graphs or Petri net) are suggested in [13] and pre-specified intermediary textual notations (TN) of BPMN are proposed in [24]. However, ordinary TNs require further conversion into diagrams (i.e., GNs) and do not allow us to see and analyze the output of a language model immediately in a user-friendly/understandable manner.

Nowadays, multiple TNs such as DOT[3], Mermaid.js[4], Penrose[5], D2[6], and PlantUML[7] exist [43]. Mermaid.js (MER) and Graphviz (GV) have been selected, because they are widely used, well-documented and thus it is reasonable to assume that LLMs might yield consistent results when generating them (see Sect. 3.1).



```
flowchart LR
   0:startevent:((startevent)) --> 1:task:(task1)
   1:task:(task1) --> 2:exclusivegateway:{x}
   2:exclusivegateway:{x} --> 3:task:(task2)
   2:exclusivegateway:{x} --> 4:task:(task3)
   3:task:(task2) --> 5:exclusivegateway:{x}
   4:task:(task3) --> 5:exclusivegateway:{x}
   5:exclusivegateway:{x} --> 6:endevent:((endevent))
```

(a) Mermaid.js

```
digraph G {
   rankdir=LR;
   node [style=filled,fillcolor=lemonchiffon];  "start_1" -> task1;
   "start_1"[shape=circle label=""];              task1 -> "seg_1";
   "end_1"[shape=doublecircle label=""];          "seg_1" -> task2;
   "seg_1"[shape=diamond label="X"];              "seg_1" -> task3;
   "meg_1"[shape=diamond label="X"];              task2 -> "meg_1";
   task1[shape=rectangle];                        task3 -> "meg_1";
   task2[shape=rectangle];                        "meg_1" -> "end_1"
   task3[shape=rectangle];                      }
```

(b) DOT Graphviz

Fig. 1: TN & GN of Text Fragment given in TF1

The TN contains the following elements: At first, we define the orientation of the graph and custom structure for all nodes and edges in it. Then, specific features are assigned for every node to represent particular BPMN element. For example, the simple process description

"After task1, either task2 or task3 are conducted." (Text Fragment TF1)

can be converted by a LLM into the TN and finally into the GN shown in Fig. 1.

Prompt Engineering: LLMs like GPT are general-purpose models performing a wide range of tasks. Usually, fine-tuning is required to allow an LLM to generate output that corresponds to the preferences of a particular user/task, making it task-specific [7]. However, fine-tuning can be computationally and storage-intensive [28], and its performance is closely connected with the amount of available training data, which could be an issue in some domains [15].

[3] https://graphviz.org/doc/info/lang.html
[4] https://mermaid.js.org/
[5] https://penrose.cs.cmu.edu/
[6] https://d2lang.com/tour/intro
[7] https://plantuml.com/

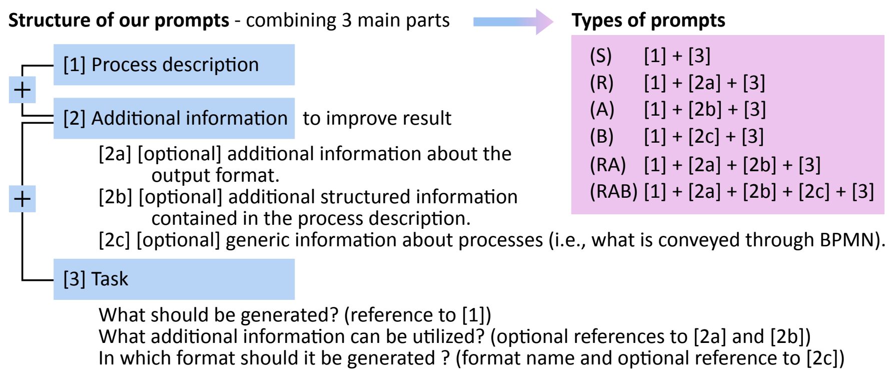

Structure of our prompts - combining 3 main parts
[1] Process description
[2] Additional information to improve result
[2a] [optional] additional information about the output format.
[2b] [optional] additional structured information contained in the process description.
[2c] [optional] generic information about processes (i.e., what is conveyed through BPMN).
[3] Task
What should be generated? (reference to [1])
What additional information can be utilized? (optional references to [2a] and [2b])
In which format should it be generated ? (format name and optional reference to [2c])
Types of prompts
(S) [1] + [3]
(R) [1] + [2a] + [3]
(A) [1] + [2b] + [3]
(B) [1] + [2c] + [3]
(RA) [1] + [2a] + [2b] + [3]
(RAB) [1] + [2a] + [2b] + [2c] + [3]

Fig. 2: Prompt Structure and Types of Prompts

Prompt engineering (i.e., the design of a prompt providing clear instructions such that LLM can perform the required task in the best and the most efficient way) can be employed to overcome these limitations [34].

Figure 2 shows how the prompts for the approach presented in this paper have been designed. Each prompt consists of three parts: [1] a process description, [2] some additional information, and [3] the actual task that should be solved by the LLM, i.e., "generate a graph" (cmp.[8]). Each part of the prompt is designed separately with the help of iterative learning, where the prompt is tested and refined continually based on the output from the previous version of a prompt [37]. By combining various parts of the optional pieces of information, such as those in [2a], [2b], and [2c] in different ways, we create six different prompts (see Fig. 2, Types of prompts). The custom user input, e.g., the process description [1], and the actual task to be performed by the LLM [3] remain unchanged. Thus, the core task, passed to the LLM remains the same. Each prompt can be considered distinct, as the additional information can greatly impact the language model response. For the experiment, we employ a zero-shot prompting strategy, providing instructions to LLMs without the use of examples, enabling them to perform tasks for which they have not been explicitly trained [37].

3 Evaluation

We evaluate the LLMs by comparing the process models generated by them to process models created by humans in terms of completeness (cf. Sect. 3.1) and correctness (cf. Sect. 3.2). Overall we use two data sets: the PET data set[9] [8] for the evaluation and an independent (real-world) data set [35] (see Sect. 3.3) to validate the results.

The PET dataset is used for process extraction from natural language text. It contains 45 textual process descriptions together with human-annotated process elements in a corpus of these process descriptions. Out of 45 process descriptions only seven examples are taken over [9], as they are of different lengths and complexity. The variety in length

[8] `https://anonymous.4open.science/r/convermod-92D5/`
[9] `https://huggingface.co/datasets/patriziobellan/PET`

and complexity is crucial, because the prompt length and content supplied to the LLM directly affect the quality of the generated response [25].

Based on the human annotations from the PET dataset, for seven selected examples, BPMN models are created. These models are to be considered as correct ground-truth process models and to be compared with the models generated by LLMs and models from the benchmark (see Fig. 3).

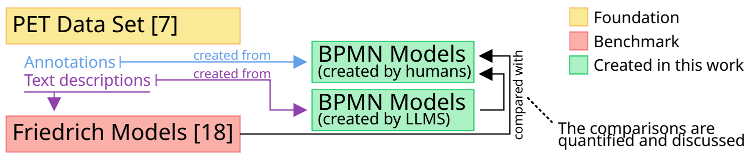


Fig. 3: Evaluation Design

For benchmarking purposes, we employ the models outlined in [21]. These models fall into two categories: BPMN models designed by a human modeler **F.Human** and those generated by an automated system **F.System** described in [21].

Despite the fact that **F.Human** models were created by human modelers, they cannot be considered as gold standard references and are used only for benchmarking for several reasons. First, many of these models were translated by the same authors from other process modeling languages without expert validation [12]. Second, human modelers tend to extend the models with the content that was not originally provided in the process description or reduce the information in the model based on their comprehension, modeling experience, domain knowledge, defined model purposes, and perspectives pursued by stakeholders [39].

For the evaluation, two basic categories of BPMN elements are considered: (a) flow objects (start and end events, tasks, exclusive and parallel gateways) and (b) connecting objects (sequence flows). The concept of merging gateways is addressed as follows: if the LLM does not generate a merging gateway, we do not consider it as an error, as a merging gateway concept is not a strict principle of the BPMN standard, and is often ignored by human modelers. If two tasks are merged into one, we consider it as a correct LLM response and split it manually into distinct tasks (e.g., "check and repair hardware" is equal to a sequence of two tasks "check hardware" and "repair hardware").

As LLMs, we choose GPT3 and GPT4, as they show promising performance [31]. Using the 6 prompts described in Sect. 2 and 7 PET examples, both GPT3 and GPT4 are instructed to create MER and GV graphs. All models (see Fig. 3) are transformed into a list of 3-tuples, where every tuple is an ordered sequence (source, connector, target). One tuple represents an edge of a model/graph (see Fig. 4).

3.1 Assessing Process Model Completeness

The following metrics are selected to quantitatively assess the generated models, i.e., the Jaccard index, precision, and recall.

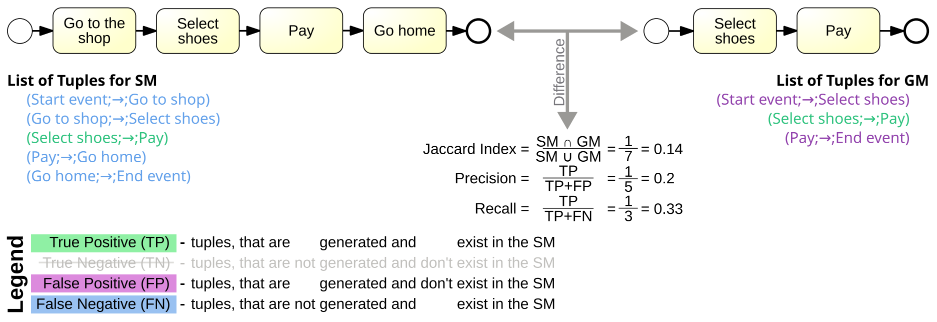

Go to the shop
Select shoes
Pay
Go home
Difference
Select shoes
Pay
List of Tuples for SM
(Start event;→;Go to shop)
(Go to shop;→;Select shoes)
(Select shoes;→;Pay)
(Pay;→;Go home)
(Go home;→;End event)
List of Tuples for GM
(Start event;→;Select shoes)
(Select shoes;→;Pay)
(Pay;→;End event)
Jaccard Index = SM ∩ GM / SM ∪ GM = 1/7 = 0.14
Precision = TP / TP+FP = 1/5 = 0.2
Recall = TP / TP+FN = 1/3 = 0.33
Legend
True Positive (TP) - tuples, that are generated and exist in the SM
~~True Negative (TN)~~ - tuples, that are not generated and don't exist in the SM
False Positive (FP) - tuples, that are generated and don't exist in the SM
False Negative (FN) - tuples, that are not generated and exist in the SM

Fig. 4: Quantitative Evaluation Metrics for Generated Models

In the NLP domain, precision and recall are used for classification problems or information retrieval evaluation [36]. We compute precision and recall metrics to assess the overall quality of the generated models.

However, precision and recall are metrics that are mostly biased towards the positive class and are not useful when used in isolation [20]. Thus, Jaccard similarity is considered as a third metric. This method can efficiently determine the proximity of two data sets without data redundancy, providing a balanced similarity measure and a straightforward indicator of how similar two sets are [38]. Additionally, it is unaffected by the size of the dataset, making it a reliable option for determining similarity when the sets are of various sizes [19]. It also allows us to determine the similarity between two sets without considering the sequence order of the items in a sample set. We use the Jaccard index to measure the similarity between the model generated by the LLM and the standard model (see Fig. 4). All parameters and formulas for the metrics are shown in Fig. 4.

In Tab. 1, the results for the Jaccard Index (JI) are presented. As we can see, by using prompts with only the process description or with the process description and additional information that does not provide sufficient instructions about the textual representation format (i.e., (S), (A) or (B)), the JI value does not reach more than 27% in the best case (20% on average).

For the rest of the cases, using GPT3, a maximum of 52% similarity is achieved. However, there is a 13% decrease between models built with (R) and (RAB) prompts. This might suggest that the more content is supplied, the worse the LLM's understanding of the instructions is. By generating models with GPT4, we attain JI values of 59, 57% and 54% with (RAB), (RA) and (R) prompts, respectively. These results are similar to those reached by the automated system and 10% higher than the result obtained by a human modeler from the benchmark.

Similar to the JI results (see Tab. 1), Tab. 2 shows that when providing less informative instructions to LLMs models, models with lower precision are created. Only between 32-45% of all generated tuples are correct. Nonetheless, models designed by a human modeler score 38% precision compared with the standard. With more content-substantial prompts, mostly 53-70% precision is reached. Hence, generated model precision is 5%-points better compared to the benchmark with the automated system and almost 2 times better compared to the human-designed models.

Table 1: Average Jaccard Index for PET standard models and textual notations (TNs) generated with GPT3 and GPT4; abbreviations see above

LLM	TN	S	A	R	RA	RAB	B	F. Human	F. System
GPT3	GV	0.03	0.11	0.44	0.47	0.15	0.03	0.47	0.58
	MER	0.18	0.22	0.52	0.47	0.39	0.19	0.47	0.58
GPT4	GV	0.13	0.17	0.50	0.56	0.46	0.40	0.47	0.58
	MER	0.14	0.22	0.54	0.57	0.59	0.27	0.47	0.58

Table 2: Average Precision for PET standard models and textual notations (TNs) generated with GPT3 and GPT4; abbreviations see above

LLM	TN	S	A	R	RA	RAB	B	F. Human	F. System
GPT3	GV	0.06	0.24	0.57	0.58	0.26	0.07	0.38	0.65
	MER	0.37	0.45	0.70	0.60	0.55	0.38	0.38	0.65
GPT4	GV	0.28	0.36	0.60	0.66	0.53	0.48	0.38	0.65
	MER	0.32	0.43	0.65	0.67	0.70	0.38	0.38	0.65

Table 3 presents recall values. In most cases, using (R),(RA) and (RAB) prompts, a recall of 51–71% is achieved. That is similar to the benchmark, where 58% and 69% recall was obtained by a human modeler and an automated system, respectively. Executing less descriptive prompts, only between 5% and 30% of target tuples were covered by generated models.

Table 3: Average Recall for PET standard models and textual notations (TNs) generated with GPT3 and GPT4; abbreviations see above

LLM	TN	S	A	R	RA	RAB	B	F. Human	F. System
GPT3	GV	0.04	0.14	0.53	0.58	0.26	0.05	0.58	0.69
	MER	0.23	0.28	0.57	0.57	0.51	0.25	0.58	0.69
GPT4	GV	0.20	0.25	0.62	0.70	0.61	0.51	0.58	0.69
	MER	0.19	0.30	0.65	0.70	0.71	0.30	0.58	0.69

Summary: In general, better results are achieved by applying prompts with more explanatory content, i.e., (R), (RA), and (RAB). Results of JI over 50% along with both recall and precision exceeding 70% can be considered satisfactory. However, there is a room for the further improvement.

We can see from Tab. 4 that utilizing MER, GPT3 is able to achieve 15–20%-points better scores compared to GV. On average, using MER, GPT3 achieves almost as good results as GPT4 showing only a 5%-points decrease in performance in contrast to GPT4. At the same time GPT4 reaches 20%points better results using GV. There is a slight difference (3%-points) between both text-based graph representations applying GPT4.

In addition, we considered the same metrics for the task extraction evaluation by taking out the set of tasks from the standard models and models created by LLMs. Despite relatively low JI values of models in general (between 21 and 39%), executing different prompts, LLMs are able to achieve 51–71% JI when comparing task sets. Recall

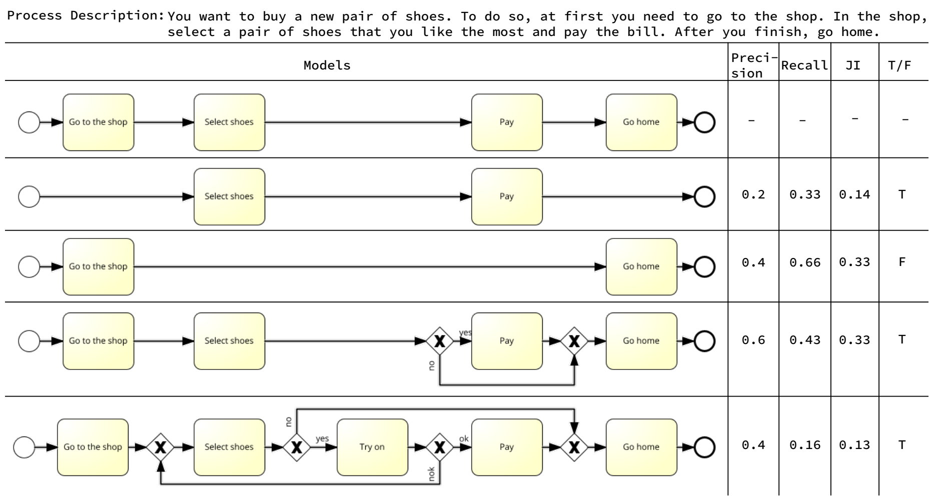
Table 4: Average of average: Jaccard Index, Recall, Precision for Models and Tasks; abbreviations see above

Metric	JI		Precision		Recall		JI		Precision		Recall	
TN	GV	MER	GV	MER	GV	MER	GV	MER	GV	MER	GV	MER
	Models						Tasks					
GPT3	0.21	0.33	0.3	0.51	0.2	0.4	0.51	0.68	0.73	0.85	0.57	0.76
GPT4	0.37	0.39	0.48	0.53	0.48	0.47	0.71	0.69	0.78	0.84	0.85	0.78

and precision for the task extraction reach up to 85% on average, and are mostly similar or even better than by the human modeler and an automated system from the benchmark (see Tab. 4 and Git repository).

3.2 Assessing Process Model Correctness

Process modeling typically results in model variability, i.e., the same process can be modeled in multiple variations from a different perspective, level of abstraction, and granularity [42]. Figure 5 depicts an example for different process models created for the same process description.



Process Description: You want to buy a new pair of shoes. To do so, at first you need to go to the shop. In the shop, select a pair of shoes that you like the most and pay the bill. After you finish, go home.

Models	Preci-sion	Recall	JI	T/F
Go to the shop → Select shoes → Pay → Go home	-	-	-	-
Select shoes → Pay	0.2	0.33	0.14	T
Go to the shop → Go home	0.4	0.66	0.33	F
Go to the shop → Select shoes → X (yes: Pay; no) → X → Go home	0.6	0.43	0.33	T
Go to the shop → X → Select shoes → X (yes: Try on; no) → X (ok: Pay; nok) → X → Go home	0.4	0.16	0.13	T

Fig. 5: Model Variability Example: Different Granularities and Perspectives

Model variability complicates model evaluation. For instance, if we consider the results presented in Tab.s 1 - 3, models designed by a human modeler have at least 10%-points lower metric values in comparison with the values gained by an automated system. The precision of a human modeler was two times worse than that of an automated system, reaching only 38%.

Low precision indicates that a human modeler tends to use another level of abstraction during process model design and tends to add further tasks, i.e., read between the lines. Thus, quantitative metrics cannot be considered objective and universal (see Fig. 5). Quantitative metrics could be used to evaluate the completeness of the model with respect to the original process description, but not its correctness.

In this work, we refer to semantic correctness, i.e., whether the generated model corresponds to its original process description. Due to model variability (see Fig. 5), we acknowledge the possibility of certain tasks being omitted or new tasks being introduced. However, this is acceptable only to the extent that it does not violate the semantic information or logical flow of the provided process description and must align with the introduced domain framework.

Hence, every model is evaluated by a modeling expert to determine whether the models generated by LLMs truly adhere to the original process descriptions. Both the process description and the process model are studied by an expert to assess whether the model covers all aspects mentioned in the text description and if semantic consistency between both representations is maintained. Based on this evaluation, a ‘true’ or ‘false’ value (1 or 0) is assigned to each model, respectively. A model is deemed correct if it captures all logical aspects of the process description.

Table 5: Correctness of models generated by GPT3 and GPT4

LLM	**TR**	**R**	**RA**	**RAB**	**F. Human**	**F. System**
GPT3	GV	2	3	2	7	4
	MER	3	2	1	7	4
GPT4	GV	6	4	2	7	4
	MER	6	6	5	7	4

Table 5 shows the average results of models generated by GPT3 and GPT4 by executing (R),(RA) and (RAB) prompts. Each row represents the number of correctly generated models out of seven selected PET process descriptions for a particular prompt.

None of the LLMs are able to achieve the same result as human modelers. GPT3 is able to generate only three correct models out of seven possible with both MER and GV representations, which is one model worse than automated systems. However, GPT4 generated six logically correct models (again, see Tab. 5). That is only one model less, when compared to a human modeler's result and 30%-points better as opposed to an automated system from the benchmark [21].

Table 6: Average Ratio of Correctly Generated Models per Case

Case	# Words	# Tasks	GPT3	GPT4	AVG
10.13	39	3	0.33	0.67	0.50
10.6	30	4	0.67	1.00	0.83
10.1	29	4	0.67	1.00	0.83
5.2	83	7	0.50	0.67	0.58
3.3	71	7	0.00	0.83	0.42
1.3	162	11	0.00	0.00	0.00
1.2	100	10	0.00	0.67	0.33

Regarding the correctness of models grouped by cases, the worst result occurred for case 1.3 in Tab. 6. In this case, none of the generated models are correct. The underlying textual description is the longest and has the highest number of tasks. The best results are achieved by both GPT3 and GPT4 for cases 10.1 and 10.6. These descriptions are the shortest. These observations suggest a correlation between the complexity of the process and the quality of the generated model. However, not only the complexity of the process is critical. Case 10.13 is one of the shortest and most simple, containing three tasks. However, 50% of the models generated for this case are incorrect.

The possible explanation for this bad performance may be poor clarity, high complexity, and bad readability of the given process description for case 10.13.

This might imply that, above all, it is essential to convey the process using straightforward and unambiguous language to enhance the comprehensibility and readability of its description in order to achieve a high-quality model.

Table 7: Number of Correctly Generated Models (NCM) for all Cases

LLM	GPT3		GPT4	
TR	GV	MER	GV	MER
NCM	7	6	12	17
Accuracy in %	33	29	50	81

Summary: Considering the maximum number of correctly generated models, both types of textual representations got the same results. Nonetheless, taking into account the number of correct generated models, GPT4 produces 17 correct models out of 21 with MER, reaching a performance of 81%. GV achieved a 57% accuracy. Using GPT3, the average accuracy of 30% is reached (see Tab. 7).

3.3 Validation

Since all prompts and their corresponding artefacts are generated using the PET dataset and a trial-and-error approach, we decide to conduct a control evaluation with our initial dataset [35], introduced in [30]. The validation set contains 21 textual process descriptions from 6 topics or domains. This is done to check whether considered prompts and artefacts can yield the optimal results with other data, or they are affected by potential biases, overfitting, or lack of generalization. In addition, due to our limited knowledge about the training process of GPT models, it is impossible to determine whether the LLM has been trained with any particular dataset already [4].

For this purpose only the best results from quantitative and qualitative evaluations, as well as survey feedback, are considered, i.e., GPT4 with MER, (R), and (RA) prompts. Every generated model is labeled as correct if it adheres to the BPMN standard and does not include any elements or relations, that contradict the original process description. In addition to correctness, we take completeness into account (see Sect. 3.1).

Despite the fact that with both (R) and (RA) the average text similarity of 74% was reached, utilizing (R) prompt, 10 models out of 21 are generated correctly, and with (RA) only 8 (see Git). The average achieved accuracy is only 43%, which is half of what was attained using the PET dataset (see Tab. 7).

Surprisingly, only 52% of all models got the same labels for the same process description, i.e., they are either both considered as correct or incorrect. The 50% variation in labeling can be attributed to the ambiguity of language, the complexity of interpreting tasks, and the model's inherent variability in processing such tasks.

4 Discussion

Considering only the completeness of a model there is no substantial difference between the performance of GPT3 and GPT4. However, looking at completeness and correctness of a model (qualitative and quantitative metrics) GPT4 performs significantly better (50%) than GPT3. Prompt engineering and the quality of input information significantly influence model completeness and correctness. A good prompt provides sufficient information about the process and its desired output format and additional details about task specifics (such as BPMN descriptions) may not always be necessary (see Sect. 3.1, 3.2).

Not only factors such as selected textual representation, complexity of the provided process and instructions, the length of textual description have an influence on the quality of a generated model, but also the clarity and the structure of the process description itself are important. The existing dependency between the quality of a generated model and its textual description (see Sect. 3.2) emphasizes that LLM models generate human-like text based on predefined patterns but may not possess true comprehension (i.e., can LLM models really 'reason' or it is still just a stochastic parrot) and leads to a reasonable question, whether LLMs can be used for specific tasks, where human understanding and experience are required.

Completeness and correctness of a model are vital characteristics, but they alone do not provide sufficient information. Generated models cannot be correct without being complete up to a certain level, and vice versa. At the same time, both completeness and correctness can be insufficient. Completeness is influenced by selected metrics, model complexity, and the chosen level of abstraction. Correctness strongly relies on the expertise of the modeling expert. The expert's background and experience determine their judgments, potentially resulting in varying interpretations and subjective assessments of what constitutes correctness.

Finally, empirical results merely represent a snapshot in time of (largely) commercial LLM capabilities. In order to assess the frontier of what is currently possible, we opted to focus our assessment on proprietary, closed LLMs (or rather: on the commercial product capabilities containing these LLMs) instead of recent, yet currently arguably weaker, openly available LLMs[10].

5 Related Work

According to [27] business process models can be generated from different sources, such as business rules, standard operating procedures, spreadsheets and unstructured text. This paper focuses on unstructured text, as not everyone understands specific formats and notations, but essentially everyone understands at least one natural language [16]. The idea of transforming unstructured text into a structured, diagram-based representation such as UML, ER diagrams, BPMN, and DECLARE, is not novel. Most existing methods rely on

[10] `https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard`

text pattern search, rule-based approaches, or semantic analysis (e.g., [23,44,21]). [40,2] propose techniques for automatic annotation of textual process descriptions utilizing supervised machine learning-based approaches. Nonetheless, due to the absence of suitable publicly available datasets containing real-life complex data, applying machine learning techniques becomes challenging [10]. Even with proprietary datasets, confidentiality requirements may not allow for the training of specialized models on much of the data that is potentially available within organizations. Hence, natural language processing (NLP) holds additional promise for research, particularly when applied in the context of utilizing Large Language Models (LLMs). [11] propose the extraction of process elements and relations using prompts with different levels of pre-knowledge and in [30] language models were utilized to extract tasks with different label length. In [24], the generation of an entire model using a specific level of abstraction is presented. [18] utilizes the JSON format to improve LLMs' capabilities to generate BPMN models, Entity-Relationship (ER), and UML Class Diagrams. There exists no methodology for creating process models that provide both LLM-compliant and graphical representation. Instead, only intermediate textual notations that must be further parsed and converted into graphical models are used. Additionally, so far, only syntactical correctness to a selected format and quantitative evaluation of the generated models have been performed, without considering their semantic correctness (how well they adhere to the provided process description). The evaluation of model correctness can offer new insights into the process of process model creation, since the ambiguity of natural language can lead to multiple interpretations of the described process, depending on the assumed contexts [1]

6 Conclusions and Research Directions

Overall, chatbots are in principle ready to be applied in practice. Yet the domain expert should always check the results, although this might still lead to variations in model quality. Additionally, the particular chatbot implementation should take into account the complex relationships between model completeness, correctness, textual process description, textual representation and prompt engineering such that the modeling capability of the chatbot can be tuned towards the respective needs of the domain expert. The results for these relationships, given the quantitative and qualitative evaluation, should be risen to the awareness of the domain expert via training before roll-out of the chatbot or added to the conversational user interface of the chatbot, e.g., by means of selecting the textual representation of the generated model. Considering that the acquisition of as-is models consumes up to 60% of the time spent on process management projects [22], chatbot-based partial automation as T2M transformation can be sufficiently impactful, even if substantial human refinement is required. Future research will focus on integrating the strong language capabilities of chatbots with the specialized capabilities of existing knowledge-based tools. The integrative research direction is more promising than chatbot training with specialized process modeling training sets featuring native process models, e.g., process models in BPMN format and a number of semantic targets, such as information on the existence of deadlocks in a process model. First, pure chatbot training ignores the vast modeling knowledge encoded in existing tools and is futile unless it serves as an intermediate step that unlocks further value.

References

1. van der Aa, H., Carmona, J., Leopold, H., Mendling, J., Padró, L.: Challenges and opportunities of applying natural language processing in business process management. In: Computational Linguistics. pp. 2791–2801 (2018)
2. Ackermann, L., Neuberger, J., Jablonski, S.: Data-driven annotation of textual process descriptions based on formal meaning representations. In: Advanced Information Systems Engineering. pp. 75–90 (2021)
3. Ali, M., Fromm, M., Thellmann, K., Rutmann, R., Lübbering, M., Leveling, J., Klug, K., Ebert, J., Doll, N., Buschhoff, J.S., et al.: Tokenizer choice for llm training: Negligible or crucial? arXiv preprint arXiv:2310.08754 (2023)
4. Arkoudas, K.: GPT-4 can't reason. CoRR **abs/2308.03762** (2023). `https://doi.org/10.48550/ARXIV.2308.03762`
5. Azevedo, L.G., Rodrigues, R.D.A., Revoredo, K.: BPMN model and text instructions automatic synchronization. In: Enterprise Information Systems. pp. 484–491 (2018). `https://doi.org/10.5220/0006809604840491`
6. Badakhshan, P.: Process modeling lifecycle – a comprehensive view derived from multiple case studies (2023), `https://blogs.sap.com/2023/09/07/process-modeling-lifecycle-a-comprehensive-view-derived-from-multiple-case-studies/`
7. Bakker, M., Chadwick, M., Sheahan, H., Tessler, M., Campbell-Gillingham, L., Balaguer, J., McAleese, N., Glaese, A., Aslanides, J., Botvinick, M., et al.: Fine-tuning language models to find agreement among humans with diverse preferences. Advances in Neural Information Processing Systems **35**, 38176–38189 (2022)
8. Bellan, P., van der Aa, H., Dragoni, M., Ghidini, C., Ponzetto, S.P.: PET: an annotated dataset for process extraction from natural language text tasks. In: Business Process Management Workshops. pp. 315–321 (2022)
9. Bellan, P., Dragoni, M., Ghidini, C.: A qualitative analysis of the state of the art in process extraction from text. In: Italian Association for Artificial Intelligence. pp. 19–30. CEUR Workshop Proceedings (2020)
10. Bellan, P., Dragoni, M., Ghidini, C.: Process extraction from text: Benchmarking the state of the art and paving the way for future challenges (2021), `https://api.semanticscholar.org/CorpusID:238531683`
11. Bellan, P., Dragoni, M., Ghidini, C.: Extracting business process entities and relations from text using pre-trained language models and in-context learning. In: Enterprise Design, Operations, and Computing. pp. 182–199 (2022)
12. Bellan, P., Ghidini, C., Dragoni, M., Ponzetto, S.P., van der Aa, H.: Process extraction from natural language text: the pet dataset and annotation guidelines. In: Workshop on Natural Language for Artificial Intelligence (2022)
13. Berti, A., Qafari, M.S.: Leveraging large language models (llms) for process mining. CoRR **abs/2307.12701** (2023). `https://doi.org/10.48550/arXiv.2307.12701`
14. Beverungen, D.: Exploring the interplay of the design and emergence of business processes as organizational routines. Bus. Inf. Syst. Eng. **6**(4), 191–202 (2014)
15. Busch, K., Rochlitzer, A., Sola, D., Leopold, H.: Just tell me: Prompt engineering in business process management. In: Enterpr., Business-Process and Inf. Syst. Modeling. pp. 3–11 (2023)
16. Dalianis, H.: A method for validating a conceptual model by natural language discourse generation. In: Advanced Inf. Syst. Engineering. pp. 425–444 (1992)
17. Doren, A., Markina-Khusid, A., Cotter, M., Dominguez, C.: A practitioner's guide to optimizing the interactions between modelers and domain experts. In: Systems. pp. 1–8 (2019)
18. Fill, H., Fettke, P., Köpke, J.: Conceptual modeling and large language models: Impressions from first experiments with chatgpt. Enterp. Model. Inf. Syst. Archit. Int. J. Concept. Model. **18**, 3 (2023)

19. Fletcher, S., Islam, M.Z.: Comparing sets of patterns with the jaccard index. Australas. J. Inf. Syst. **22** (2018), `https://journal.acs.org.au/index.php/ajis/article/view/1538`
20. Foody, G.M.: Challenges in the real world use of classification accuracy metrics: From recall and precision to the matthews correlation coefficient. PLoS One **18**(10), e0291908 (Oct 2023)
21. Friedrich, F.: Automated Generation of Business Process Models from Natural Language Input. Master's thesis, Humbold-University zu Berlin (2010)
22. Friedrich, F., Mendling, J., Puhlmann, F.: Process model generation from natural language text. In: Advanced Information Systems Engineering. pp. 482–496 (2011)
23. Ghose, A., Koliadis, G., Chueng, A.: Process discovery from model and text artefacts. In: Services Computing Workshops. pp. 167–174 (2007)
24. Grohs, M., Abb, L., Elsayed, N., Rehse, J.: Large language models can accomplish business process management tasks pp. 453–465 (2023). `https://doi.org/10.1007/978-3-031-50974-2_34`
25. Gu, X., Yoo, K.M., Lee, S.: Response generation with context-aware prompt learning. CoRR **abs/2111.02643** (2021), `https://arxiv.org/abs/2111.02643`
26. Gutschmidt, A., Lantow, B., Hellmanzik, B., Ramforth, B., Wiese, M., Martins, E.: Participatory modeling from a stakeholder perspective: On the influence of collaboration and revisions on psychological ownership and perceived model quality. SoSyM **22**, 1–17 (2022)
27. Honkisz, K., Kluza, K., Wisniewski, P.: A concept for generating business process models from natural language description. In: Knowledge Science, Engineering and Management. pp. 91–103 (2018)
28. Hu, Z., Wang, L., Lan, Y., Xu, W., Lim, E., Bing, L., Xu, X., Poria, S., Lee, R.K.: LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In: Empirical Methods in Natural Language Processing. pp. 5254–5276 (2023)
29. Kannengiesser, U., Oppl, S.: Business processes to touch: Engaging domain experts in process modelling. vol. 1418 (09 2015)
30. Klievtsova, N., Benzin, J., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modelling: State of the art, applications, and implications in practice. In: Business Process Management Forum. pp. 319–336 (2023)
31. Klievtsova, N., Benzin, J., Kampik, T., Mangler, J., Rinderle-Ma, S.: Conversational process modelling: State of the art, applications, and implications in practice. CoRR **abs/2304.11065** (2023). `https://doi.org/10.48550/ARXIV.2304.11065`
32. Leopold, H., van der Aa, H., Pittke, F., Raffel, M., Mendling, J., Reijers, H.: Searching textual and model-based process descriptions based on a unified data format. Software and Systems Modeling **18** (04 2019). `https://doi.org/10.1007/s10270-017-0649-y`
33. Leopold, H., Mendling, J., Polyvyanyy, A.: Supporting process model validation through natural language generation. IEEE Trans. Software Eng. **40**(8), 818–840 (2014)
34. 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. ACM CSUR **55**(9) (2023)
35. Mangler, J., Klievtsova, N.: Dataset: Textual Process Descriptions and Corresponding BPMN Models (2023). `https://doi.org/10.5281/zenodo.7783492`
36. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press, Cambridge, Massachusetts (1999), `http://nlp.stanford.edu/fsnlp/`
37. Marvin, G., Hellen Raudha, N., Jjingo, D., Nakatumba-Nabende, J.: Prompt Engineering in Large Language Models, pp. 387–402 (01 2024). `https://doi.org/10.1007/978-981-99-7962-2_30`
38. Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S.: Using of jaccard coefficient for keywords similarity. `https://api.semanticscholar.org/CorpusID:14040055`

39. Pinggera, J., Zugal, S., Furtner, M., Sachse, P., Martini, M., Weber, B.: The modeling mind: Behavior patterns in process modeling. In: Bider, I., Gaaloul, K., Krogstie, J., Nurcan, S., Proper, H.A., Schmidt, R., Soffer, P. (eds.) Enterprise, Business-Process and Information Systems Modeling - 15th International Conference, BPMDS 2014, 19th International Conference, EMMSAD 2014, Held at CAiSE 2014, Thessaloniki, Greece, June 16-17, 2014. Proceedings. Lecture Notes in Business Information Processing, vol. 175, pp. 1–16. Springer (2014). `https://doi.org/10.1007/978-3-662-43745-2_1`, `https://doi.org/10.1007/978-3-662-43745-2_1`
40. Qian, C., Wen, L., Kumar, A., Lin, L., Lin, L., Zong, Z., Li, S., Wang, J.: An approach for process model extraction by multi-grained text classification. In: Advanced Information Systems Engineering. pp. 268–282 (2020)
41. Shaw, M.L., Woodward, J.B.: Modeling expert knowledge. Knowledge Acquisition **2**(3), 179–206 (1990)
42. Smirnov, S., Reijers, H.A., Weske, M., Nugteren, T.: Business process model abstraction: a definition, catalog, and survey. Distributed Parallel Databases **30**(1), 63–99 (2012)
43. Whatley, D., Goldman, M., Miller, R.C.: Snapdown: A text-based snapshot diagram language for programming education. In: Visual Languages and Human-Centric Comp. pp. 1–9 (2021)
44. Yue, T., Briand, L.C., Labiche, Y.: An automated approach to transform use cases into activity diagrams. In: Modelling Foundations and Appl. pp. 337–353 (2010)