Revisiting Business Process Analysis through the lens of Large Language Models: Prompting experiments with BPMN process serializations

Damaris Dolha[a], Ana-Maria Ghiran[a] and Robert Andrei Buchmann[a]

Babeş-Bolyai University, Faculty of Economics and Business Administration, 58-60 T. Mihali St.,
Cluj-Napoca 400591, Romania

E-mails:
[a] *damaris.dolha@econ.ubbcluj.ro,*
[b] *anamaria.ghiran@econ.ubbcluj.ro,*
[c] *robert.buchmann@econ.ubbcluj.ro*

Abstract. Recent position papers have proposed that the traditional BPM (Business Process Management) lifecycle must be revisited considering generative AI advances, specifically by investigating how LLMs (Large Language Models) can assist various phases of the lifecycle. Inspired by that call to action, this paper reports on a series of experiments on how OpenAI's GPT 4 responds when querying the content of BPMN diagrams, as potential support for the Analysis phase of the BPM lifecycle. We are particularly interested in how BPMN content - typically available in enterprises that adopted the BPM lifecycle - should be exposed to LLM services, therefore we comparatively experiment with diagrams provided as XML serializations or as tool-specific RDF serializations. This is a comparison between a standard serialization characterized by intricate cross-referencing that compensates for the XML rigid hierarchical structure and the "semantic graph" view of RDF that is open-ended in terms of semantic annotation and can be serialized as sequences of statements that resemble natural language. The quality of the answers is assessed using the RAGAs (Retrieval-Augmented Generation Assessment) framework.

Keywords: BPMN, Process serializations, Generative AI, RDF, XML, RAGAs

1. Introduction

The paper reports on continuous experimentation efforts with how selected Large Language Models (LLMs) services interpret Business Process Model and Notation (BPMN) models exposed in a variety formats – as standard XML serializations or as non-standard tool-specific RDF graphs. This work can inform how we engage with an AI-powered version of the Business Process Management (BPM) lifecycle [1], leveraging augmentations made possible by LLM services for certain phases of the lifecycle.

For the experiments reported in this paper, the GPT 4 model of OpenAI accessed via its API was the targeted service

BPMN remains the dominant standard for documenting processes, predominantly relying on XML for diagram serializations, a variety of schemas being available for this – e.g., XPDL, BPEL-WS, BPMN XML, ADOXML. A large diversity of process-aware systems or engines [2] have been built on such XML schemas to parse process descriptions. However, in the context of semantics-driven engineering formulated by [3] the requirement of process understandability becomes just as relevant as process instantiation and execution; moreover, process instantiation must sometimes be contextualized by a semantic layer available in the form of knowledge graphs. Towards such needs, the introduction of RDF (Resource Description Framework) encoding of BPMN content brings new semantics-oriented capabilities, although such representations are not yet widely adopted and are tool-specific: some examples are the Bee-Up modeling tool [4] and various demonstrators reported by the literature [5].

Our research explores the use of XML and RDF serializations of BPMN with generative Artificial Intelligence (GenAI). The experiments harness the LLMs provided by OpenAI - in this report, GPT 4. By targeted prompts, we probe various aspects of selectively designed BPMN exemplars, to see how RDF compares with XML in facilitating interpretation of process descriptions, despite it not providing a standard vocabulary such as those available to the XML schemas for process descriptions. The adoption of knowledge graphs as process storage and BPMN knowledge structures is still an experimental proposition in artifact-oriented literature [5,6], but it allows us to explore the potential of RDF to facilitate dialogue for process analysis via generative AI, beyond the XML export treatment of having BPMN as a closed world markup data structure. Consequently, we formulate the following research question pursued by the experimentation reported in this paper:

RQ1: How does the standard BPMN XML export compare with the RDF export available in the Bee-Up modeling tool, when exposed to an LLM service tasked with answering natural language queries on process descriptions? The queries must target both the identification of construct types and the navigation of chains of relationships in realistic complex examples, complete with labels and involving interlinked diagrams (i.e. subprocess).

RQ2: What are the outcomes of this comparison for a diversity of minimalist BPMN patterns involving: (a) the most used BPMN constructs – tasks, events, gateways, pools - cf. recent literature investigating the prevalence of BPMN constructs in real collections of process descriptions [38]; (b) non-explicit labels to prevent the LLM from extrapolating narratives from the textual tokenized labels without looking at the process structure itself.

The LLM used for the experimentation reported in this paper is OpenAI's GPT 4, as the focus is now on varying the process description format. Future works will also vary the LLM services relevant for these research questions.

This research aligns with a current stream of investigation into the capability of AI to engage with conceptual models [7]. It also furthers the investigation into how LLM services can process serialized diagrammatic models, building upon initial prompting strategies that we have reported in a conference scope in [8]. The comparative analysis in this version also extends across a more diverse set of scenarios, thereby expanding the representativeness and insights of the findings, and provides more detailed experimentation design.

We focus on process serializations, rather than images, because most BPMS (Business Process Management Systems) or BPA (Business Process Automation) platforms and services rely on serializations for process repositories and model interchange, as they require deterministic interoperability or execution. The introduction of image recognition capabilities by the OpenAI's GPT-4 model significantly enhances the multi-modal interaction possibilities [9], enabling a more holistic approach to understanding and interacting with BPMN models. However, visualization cannot fully grasp a process description – many details relevant to process analysis do not manifest on a visual level (e.g., data attributes and links between diagrams).

The paper is organized as follows: in Section 2, we establish the problem scope in the context of the BPM lifecycle based on our reading of recent literature. Next, Section 3.1 explains the structural and syntactical differences between XML and RDF, Section 3.2 summarizes the model exemplars used in the experiments and Section 3.3 details the experimental setup. The core findings of our experimental study, along with our interpretation are reported in Sections 4.1 and 4.2, where the first analysis is on a full complex example and the latter focuses on minimalist BPMN patterns that are left unexplicit to force the LLM to develop its interpretation independently of the textual labeling found in diagrams. Section 5 synthesizes the results and connects them to broader implications. Concluding this study, Section 6 summarizes the findings and maps out directions for future exploration.

2. Large Language Models and the BPM lifecycle

Motivated by the need to revisit the BPM lifecycle through the lens of the capabilities of LLMs [10], our investigation checks for the suitability of LLMs - in this paper OpenAI's GPT 4 model - for the semantic querying of procedural knowledge available in BPMN formats. The process analysis phase of the BPM lifecycle has traditionally relied on process queries by various means developed over the years: graph queries [5,6], formal languages [11], visual grammars [12]. Retrieval of information from BPMN models can of course be performed by visual analysis due to the diagrammatic nature of the content, however process queries are called to provide automatable mechanisms that can also be executed over process repositories. On the other hand, in our work we use smaller, even minimalist examples, to assess the LLM suitability as a replacement for process querying – therefore no repository scalability concerns are raised in this study, as we are focusing on isolating certain workflow patterns and commonly used BPMN constructs.

The phases of the BPM lifecycle revisited in relation to LLMs radically depart from the conventions of pre-GPT process analysis tools. Recent works [10] draw attention to how these models can redefine how we think about the phases of the BPM lifecycle: in the *process identification* phase, LLMs cut through the clutter of unstructured data: they do not just find information, they can gather workflow knowledge – moving into *process discovery*, the influence of GenAI can enhance process mining frameworks. Traditionally tethered to XML event logs, RDF can push graph-based process mining [13]. When it comes to *querying processes*, where this paper's focus lies, the traditional XML frameworks and XPath/XQuery-based retrieval are now over-hauled by multi-modal AI [14] with their computer vision capabilities. Yet, reliance on computer vision has its limits and must be complemented by a semantic serialization layer to also expose non-visual aspects that remain semantically relevant – e.g., links between different models (such as RACI responsibilities on task level) or data attributes (e.g., task costs). In the *redesign* phase, AI can advocate changes, using code generation on serializations to apply workflow updates. As those updates are implemented, LLMs enrich user interactions with detailed explanations, shifting emphasis from static workflow sequences to dynamic conversational choreographies. During the *monitoring* phase, LLMs are not confined to data display, as they can interpret and analyze data.

The current report focuses on process querying and interpretation, crucial stages in process analysis, that require question-answering mechanisms and the capacity to draw inferences from the process semantics. As AI pushes the boundaries of BPM, recent technological proposals change the way processes can be analyzed. The BPMN2KG initiative [15] illustrates the conversion of BPMN 2.0 models – typically expressed in XML – into RDF-based knowledge graphs and marks a different view on the instantiation of process models. Similarly, another conversion tool [6] allows BPMN XML formats to morph into Neo4J labeled property graphs (LPG), providing an alternative graph representation that adheres to the BPMN 2.0 vocabulary.

The work in [16] laid an early foundation by exploring process querying methods and applications in BPM, establishing benchmarks that later studies would build upon. Furthermore, [12] ventured into the application of LLMs for textual analyses within BPM, demonstrating that GPT-4 can effectively derive both imperative and declarative process models from natural language descriptions – a significant advance for process querying. Shortly thereafter, analyzing the effectiveness of ChatGPT in generating and deciphering diverse conceptual models, [7] suggested certain operational nuances that influence model interpretation. Building on these foundational insights, [17] extends earlier research by not only confirming the viability of LLMs for textual analysis in BPM, but also by operationalizing these capabilities into a comprehensive process modeling framework. It takes a further step by automating the transformation of textual descriptions into standardized process models, such as BPMN and Petri nets.

Another investigation [18], this time on prompt engineering for BPM, underscored the critical importance of prompt design for obtaining consistent and reliable outputs from LLMs. Similarly, [19] reinforces the idea that prompt customization is key for enhancing BPM outcomes and expanding domain knowledge coverage, with LLMs suggesting missing concepts from business process models (serialized using XML, XPDL and XMI) and converted into CSV files for further processing. Complementing these developments, recent research [20] explores whether LLMs can substitute for domain experts by evaluating enterprise models and highlights that measuring certain quality aspects – such as completeness – remains challenging due to the inherent subjectivity and variability in certain models.

Expanding the scope further, [21] scrutinized the intersection of GPT technology with RPA, drawing attention to potential security and compliance challenges as enterprises increasingly incorporate AI-driven process automation. The integration of GPT-3.5 Turbo with Lean Six Sigma 4.0 methodologies [22] revealed promising avenues for upgrading customer service and enhancing real-time decision-making processes. Moreover, [23] proposed a forward-thinking strategy that involves merging LLMs with knowledge graphs, an approach that is particularly promising when these graphs are designed around diagrammatic procedural knowledge effectively serialized as semantic networks.

While extensive research has concentrated on the performance of LLMs on unstructured text (such as generating and refining business process models from textual descriptions [34] or extracting process information from text [35]), their ability to query structured serializations – specifically, RDF and XML serialization of BPMN process descriptions – remains largely underexplored.

This gap is significant, given the essential role that structured process representations play in BPM – traditionally, as storage format for process automation and analysis engines (hence the XML dominance) but now more relevant as semantically open-ended knowledge assets (hence our focus on RDF). Even though prior work [36] examined the performance of LLMs on unstructured formats such as tabular data, data-oriented scenarios differ significantly from those posed by structured BPMN representations (e.g., accurately parsing complex workflow patterns or managing intricate cross-references).

3. Experimental Setup

At the forefront of our study is Bee-Up 1.7 [4], a core component of the OMiLAB Digital Innovation environment [24], known for its role in semantically enriching both standard-based and domain-specific models and languages, allowing the exploitation of inter-model links as semantic bridges between a diversity of modeling languages – e.g., BPMN, UML, DMN, EPC and Petri Nets.

SAP Signavio [25] was used as a representative for the tools providing the standard BPMN 2.0 XML serialization.

Both tools provide image exports as PNG and other image formats, however the experimentation reported in this paper focuses on the XML vs RDF exports comparison, for the reasons already explained in the previous section.

3.1. Serialization comparison

The structural differences between the XML and RDF serializations are briefly explained here based on the sample diagram in Figure 1, which shows a minimal subprocess linked to a main process with pool and lanes containment, as well as labeled connectors outgoing from a gateway.

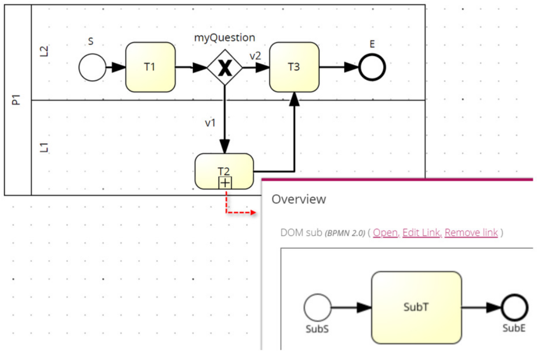
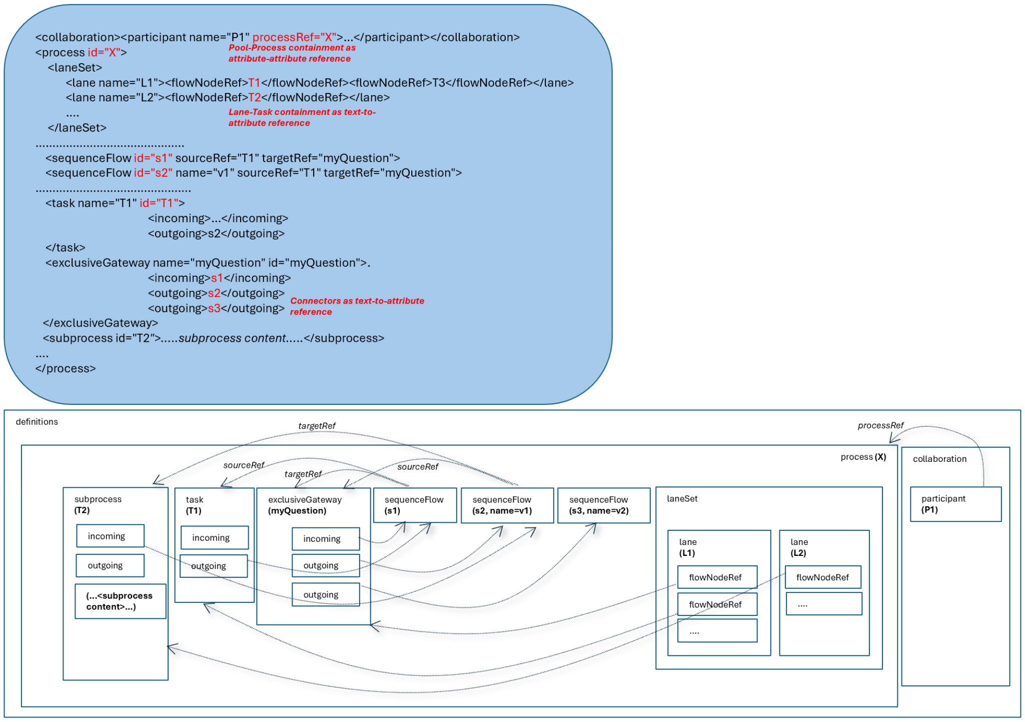


Fig. 1. BPMN exemplar comprising a link between a main process and a minimalist subprocess.

Figure 2 provides an overview of the hierarchical, DOM-based structure available in the standard serialization, conforming the standard BPMN 2.0 XML schema available in Signavio.

Because the hierarchical decomposition does not reflect the visually directed graph structure of the diagram, an intricate network of cross-references across XML tags must preserve all relevant relationships. Some of these are *attribute-attribute matches* (e.g., to specify containment between a pool and the process inside it), others are *attribute-tag matches* (e.g., to specify containment between a lane and its flow elements, but also visual connectors referencing their connector heads), *one-to-one* (e.g., an arrow can have one starting point and one ending point) or *one-to-many matches* (e.g., a gateway has multiple outgoing connectors). Only a few relationships are expressed by the implicit *parent-child XML nesting* that would be familiar to a basic parser reading such content sequentially: a process containing lanes and a subprocess containing its contents. XML parsers navigate such complex cross-references as prescribed by the governing XML schema, but process interpretation by a linear token-based parser reading it as textual content requires frequent back-and-forth jumps based on heterogeneous matching rules.



```
<collaboration><participant name="P1" processRef="X">...</participant></collaboration>
<process id="X">
   <laneSet>
        <lane name="L1"><flowNodeRef>T1</flowNodeRef><flowNodeRef>T3</flowNodeRef></lane>
        <lane name="L2"><flowNodeRef>T2</flowNodeRef></lane>
        ....
   </laneSet>
............................................
  <sequenceFlow id="s1" sourceRef="T1" targetRef="myQuestion">
  <sequenceFlow id="s2" name="v1" sourceRef="T1" targetRef="myQuestion">
.............................................
  <task name="T1" id="T1">
                          <incoming>...</incoming>
                          <outgoing>s2</outgoing>
  </task>
  <exclusiveGateway name="myQuestion" id="myQuestion">.
                          <incoming>s1</incoming>
                          <outgoing>s2</outgoing>
                          <outgoing>s3</outgoing>
 </exclusiveGateway>
 <subprocess id="T2">.....subprocess content.....</subprocess>
....
</process>
```

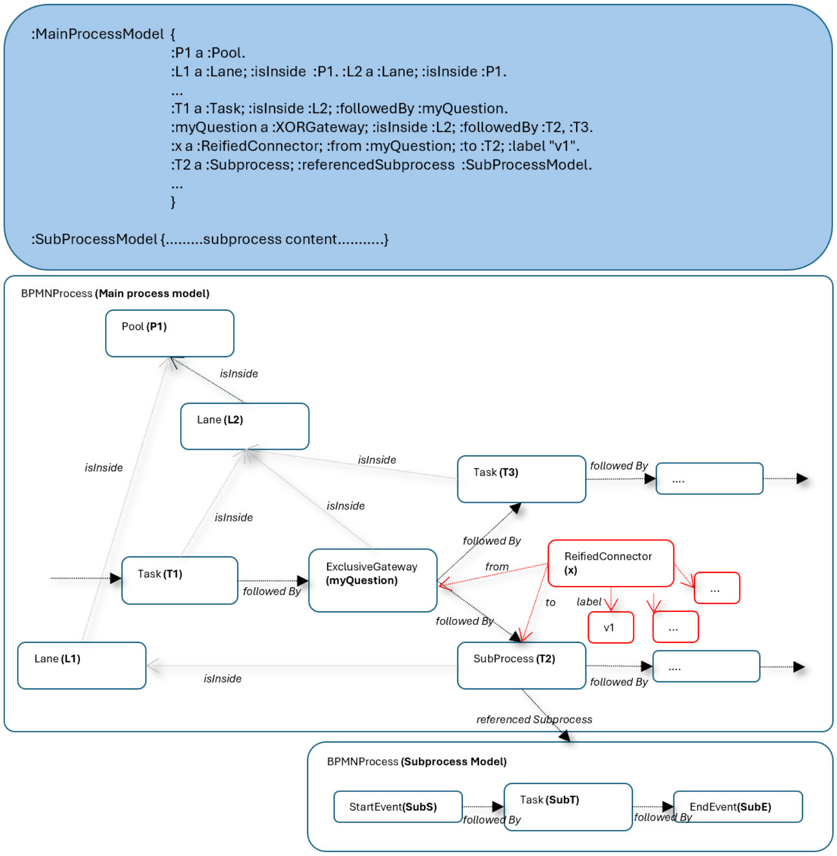
Fig. 2. Insights in the DOM-based structure of the standard BPMN XML serialization (based on the exemplar in Figure 1).

Comparatively, Figure 3 showcases the graph structure of an RDF serialization in Bee-Up, derived from transformation patterns initially formulated in [26].

The graph edges represent predicates that can be derived from various ways in which diagrammatic descriptions can express relationships: visual connectors (i.e. sequence flows, message flows), hyperlinks (e.g., links to subprocesses), containment relationships (to lanes, pools), data attributes editable in the tool (e.g., simulation attributes, connector annotations) and also open-ended properties that can be attached as annotations to any diagrammatic element (not used in this example).

Some visual connectors must emulate the "property graph" approach – i.e. graph edges having their own properties – therefore the outgoing arrows from the XOR gateway, which need to be labeled differently are highlighted as a reification pattern (the Bee-Up export does not currently employ RDF-star [40] which could simplify this further). The connector is reified to hold any attributes that are distinctively set for each instance of that connector. SPARQL rules and filters can be used to conveniently query the connectors in either the simple (non-annotated) form, or in the reified (annotated) form, depending on the process query requirements.

Subprocesses are isolated as separate named graphs, however, linked within the same RDF dataset. Leveraging such patterns, an RDF export is available in the Bee-Up modeling tool, with some terminological aspects not detailed here (tool-specific namespaces). On the top of the figure, TriG/Turtle statements show the process as it is serialized and grouped by the diagram graph where the statements belong. Many other attributes can be exported, not visible here as they are not relevant to semantically-oriented process queries (e.g., visual position) and are also filtered out by our component that delivers the graphs to GPT-4.



```
:MainProcessModel {
                    :P1 a :Pool.
                    :L1 a :Lane; :isInside  :P1. :L2 a :Lane; :isInside :P1.
                    ...
                    :T1 a :Task; :isInside :L2; :followedBy :myQuestion.
                    :myQuestion a :XORGateway; :isInside :L2; :followedBy :T2, :T3.
                    :x a :ReifiedConnector; :from :myQuestion; :to :T2; :label "v1".
                    :T2 a :Subprocess; :referencedSubprocess  :SubProcessModel.
                    ...
                    }

:SubProcessModel {.........subprocess content..........}
```

Fig. 3. Insights in the graph structure of the RDF serialization (based on the exemplar in Figure 1).

3.2. Model exemplars used in experiments

We focused our experiments on two kinds of BPMN models: (1) a realistic model of a main process linked to a subprocess, illustrated in Figure 4; (2) a set of minimalist process patterns (Figures 5-6) that are labeled non-explicitly to allow us to probe process structure interpretation while avoiding any chance of hallucinating a business scenario narrative out of textual labels. We only showcase in Figures 4-6 the Bee-Up variants, having equivalent elements to the Signavio variants.

The realistic process model in Figure 4 depicts the logic of an RPA bot planned to mimic the human actions for online shopping, as well as the data requirements for the bot to accomplish such a task and the “human in the loop” interactions – including credentials needed to perform some authentication steps on behalf of the human.

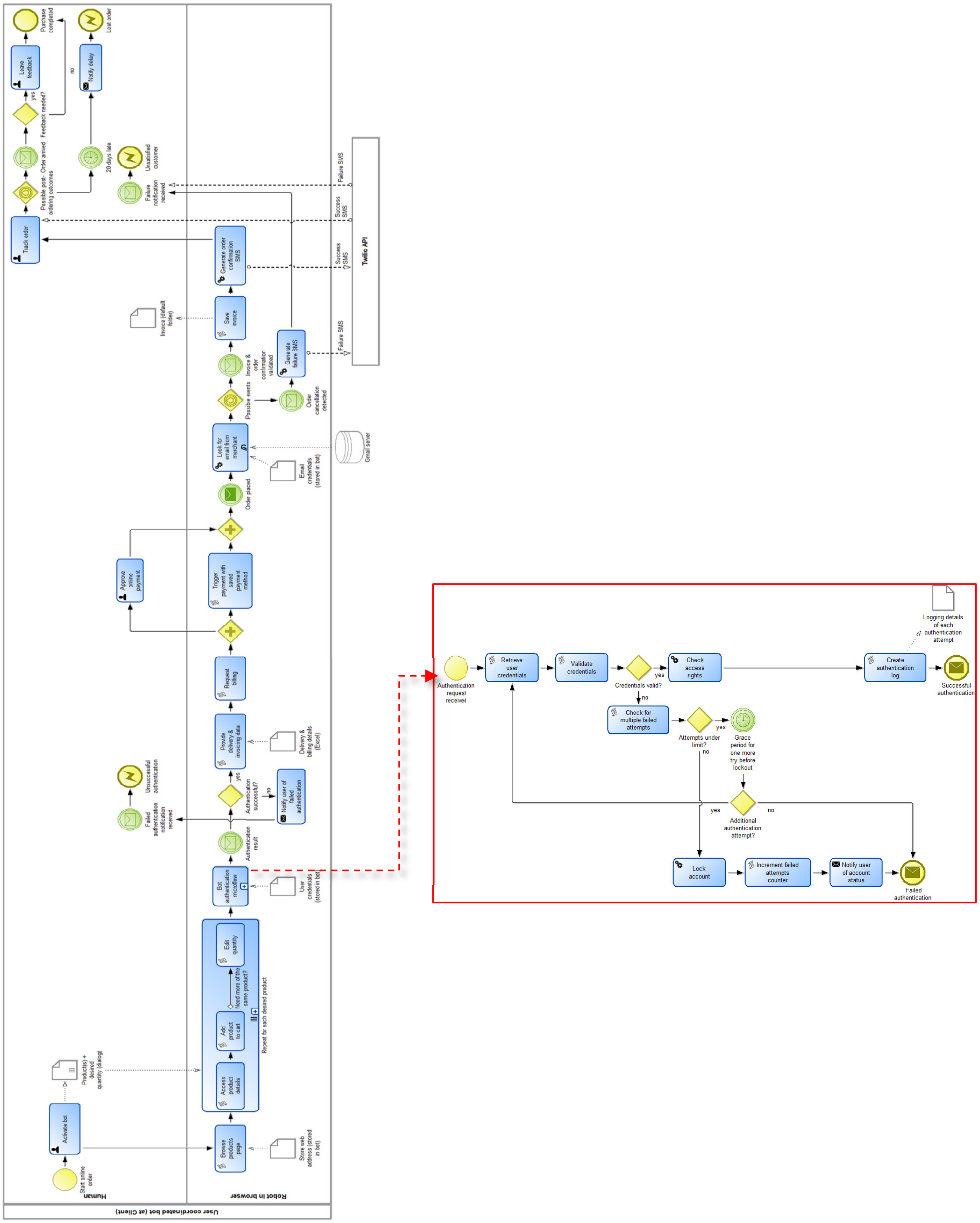
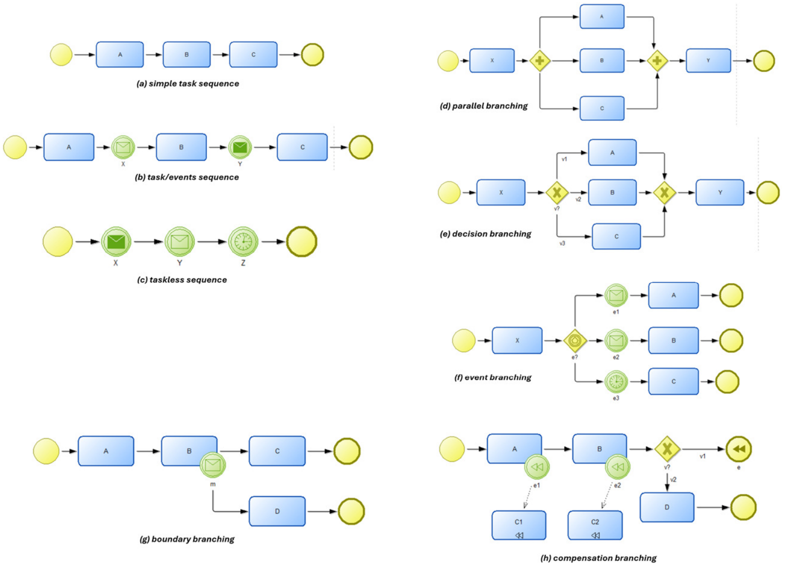

User coordinated bot (at Client)
Human
Robot in browser
Start online order
Activate bot
Product(s) + desired quantity (dialog)
Browse products page
Store web address (stored in bot)
Access product details
Add product to cart
Need more of the same product?
Edit quantity
Repeat for each desired product
Bot authentication microflow
User credentials (stored in bot)
Authentication result
Authentication successful?
yes
no
Notify user of failed authentication
Failed authentication notification received
Unsuccessful authentication
Provide delivery & invoicing data
Delivery & billing details (Excel)
Request billing
Trigger payment with saved payment method
Approve online payment
Order placed
Look for email from merchant
Email credentials (stored in bot)
Gmail server
Possible events
Order cancellation detected
Generate failure SMS
Invoice & order confirmation validated
Save invoice
Invoice (default folder)
Generate order confirmation SMS
Failure SMS
Success SMS
Twilio API
Track order
Failure notification received
Unsatisfied customer
Possible post-ordering outcomes
20 days late
Notify delay
Lost order
Order arrived
Feedback needed?
Leave feedback
Purchase completed
Authentication request received
Retrieve user credentials
Validate credentials
Credentials valid?
Check access rights
Create authentication log
Logging details of each authentication attempt
Successful authentication
Check for multiple failed attempts
Attempts under limit?
Grace period for one more try before lockout
Additional authentication attempt?
Lock account
Increment failed attempts counter
Notify user of account status
Failed authentication

Fig. 4. BPMN diagrams depicting the main BPMN process and the “Bot authentication microflow” subprocess (Bee-Up 1.7).

The examples in Figures 5-6 follow a different strategy: only generic labels are visible, forcing the LLM to look into the process structures instead of extrapolating scenario narratives inspired by labels, as this was detected as an occasional behavior on label-rich diagrams.

This also helps us assess the structural and flow-based "reasoning" abilities that may manifest, driven by the different types of connectors and the semantics they carry – sequence flows, message flows, data associations.



(a) simple task sequence

(b) task/events sequence

(c) taskless sequence

(d) parallel branching

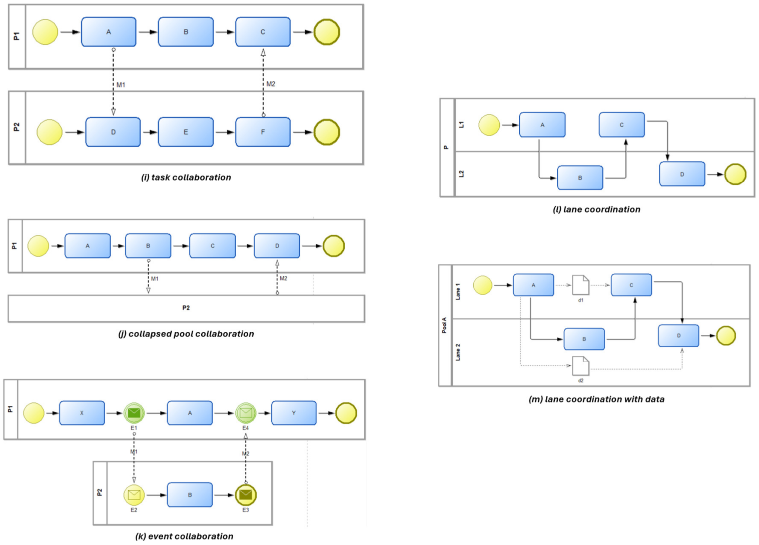
(e) decision branching

(f) event branching

(g) boundary branching

(h) compensation branching

Fig. 5. Minimalist process modeling patterns in BPMN.



(i) task collaboration

(j) collapsed pool collaboration

(k) event collaboration

(l) lane coordination

(m) lane coordination with data

Fig. 6. Collaborative and coordinated process patterns in BPMN.

3.3. Experimental framework and GPT model integration

The key feature leveraged in Bee-Up is the RDF export, which builds on previously introduced diagram-to-named graphs transformation patterns [26]. To parse and filter the RDF outputs as semantic graphs, we turned to Ontotext GraphDB 10.8 [27] after stripping away irrelevant attributes from the RDF metadata.

Ontotext GraphDB expands the standard SPARQL querying capabilities with OpenAI-oriented functions, such as *gpt:ask()*[1] that engages directly with the selected GPT model (specifically for this work, GPT-4) while exposing to it a convenient subgraph extracted from the RDF repository. The structure of the utilized query is shown below, filtering specific types of nodes and edges that are relevant for the process description (and not for its visualization):

```
# prefixes removed for concision
# <…> are placeholders for relevant graph edges or node types, as well as the user
prompt

SELECT ?answer ?rdfSer WHERE {
       SELECT (helper:rdf(helper:tuple(?x, ?prop, ?o)) AS ?rdf)
       WHERE {
              GRAPH :<graph_name> {
                     ?x a ?type ; ?prop ?o .
                     FILTER(?type IN (<RDF_class_identifier1>,
                                   <RDF_class_identifier2>,
                                   <RDF_class_identifier3>, …))
                     FILTER (?prop IN (<property1>, <property2>, <property3>, …))
              }
       }
}
BIND(helper:serializeRDF(?rdf) as ?rdfSer)
?answer gpt:ask (“prompt_template+user_question” ?rdf)}
```

We use zero-shot learning out of the prompting techniques discussed by the literature [28,29] – where the GPT model is prompted to perform a task without being provided any prior examples – and incorporate a “persona” prompt template. Specifically, in the SPARQL query, we pass this prompt template to the *gpt:ask()* function along with the actual question. This strategy defines GPT-4’s role as a question-answering assistant and sets a strict format for its responses, keeping them brief and anchored in the provided content (namely, the semantic triples). This minimizes the ambiguity, possible hallucinations and variability in the generated outputs, so the evaluation of these responses is simplified. Importantly, while our instructions clearly define the style and format of the generated response, they refrain from any explanations of the retrieved content, allowing GPT-4 to independently derive its meaning.

In order to boost performance on knowledge-intensive tasks, the authors of [30] introduced a Retrieval Augmented Generation framework that pairs a retriever with a generative model that injects relevant external context into the responses of the LLM, a concept we also leverage in order to efficiently analyze processes serialized in BPMN XML exported from Signavio. Firstly, we deploy a local Weaviate[2] container via Docker[3] to serve as our vector database, which integrates smoothly with LangChain[4] whose components we use to load the BPMN XML file. We utilize the same prompt template – used to query

---

[1] https://graphdb.ontotext.com/documentation/10.8/gpt-queries.html#gpt-ask-retrieve-a-single-answer

[2] https://weaviate.io

[3] https://weaviate.io/developers/weaviate/installation/docker-compose

[4] https://www.langchain.com

the graph database – and integrate the retriever, the prompt, as well as the GPT model into this RAG pipeline. For every query, the code invokes the RAG chain to generate an answer and simultaneously retrieves the most relevant context documents via the retriever. These answers and their associated contexts are then stored in separate lists for later evaluation.

Regarding GPT-4's configuration, we maintained a consistent setup across all platforms, including both GraphDB and our Python scripts[5]:

```
Temperature: 1
Embedding model: text-embedding-ada-002⁶
Max. tokens: 8192 tokens
Top_p: 1
Frequency and Presence penalty: 0
```

4. Evaluation and Experimental Outcomes

During April-July 2024, followed by revision refinements during October-December 2024, we examined responses produced by GPT-4 to a variety of prompts probing its BPMN-based process analysis capabilities. For clarity, we refer to the RDF variant by "Case I" and to SAP Signavio exports by "Case II". This approach is not intended to highlight limitations within any specific tool, but to explore the differences and potential insights that different serialization formats may reveal.

In the preliminary stages of our experiments, the tools deployed in the aforementioned cases demonstrated adeptness in responding to straightforward inquiries, such as identifying participants within processes or listing the sequence of steps in the provided processes. These queries, predominantly aiming at recognizing basic BPMN elements, established a baseline of competence.

Using the Retrieval Augmented Generation Assessment (RAGAs) framework [31], we compare the answers generated by the LLM from both cases against a manually crafted reference/ground truth, focusing on four key metrics: *Response Relevancy*[7], *Factual Correctness*[8], *Answer Semantic Similarity*[9] and a modified version of the *Faithfulness*[10] metric. *Response Relevancy*, initially known as "Answer Relevance", measures how well the generated answer addresses the question posed in each experiment. *Factual Correctness* ("Answer Correctness") evaluates the alignment between the generated answer and the ground truth (which is the gold standard against which the generated answers are compared). *Answer Semantic Similarity*, previously known as "Answer Similarity", focuses on how closely the meaning of the response matches that of what is expected in the ground truth. Unlike *Factual Correctness*, this metric first converts both the generated answer and the ground truth into vectors, then computes the cosine similarity between them to assess how closely they align. On top of these metrics, we updated *Faithfulness* to get a better sense of how well the generated answer matches the retrieved context. Normally, *Faithfulness* measures the factual consistency between an answer and its context. But since our context is structured (RDF triples and XML), while the generated answers are unstructured (textual descriptions) – each format organizing and representing information differently – comparing them directly can become challenging. To address this, we use OpenAI's *text-embedding-ada-002* embedding

---

[5] https://github.com/Damarissss/structured-text_RAG-pipeline_RAGAs/tree/main

[6] https://openai.com/index/new-and-improved-embedding-model/

[7] https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/answer_relevance/

[8] https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/factual_correctness/

[9] https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/semantic_similarity/

[10] https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/faithfulness/

model to convert both the answer and the retrieved structured context into vector embeddings, then compute the cosine similarity between them and average the results (rounded to two decimals) – much like how *Answer Semantic Similarity* is calculated. This new approach, which we call “Semantic Faithfulness”, offers a clearer measure of accuracy for these specific analyses.

The evaluation process for the BPMN models serialized in RDF began by loading the Turtle-serialized process description and filtering out triples that hold to no semantic relevance (visual properties). The context is paired with a specific question, its corresponding *ground truth* and GPT-4’s answer. An evaluation dataset is thus constructed from these components. Subsequently, evaluation metrics from the RAGAs framework, along with a custom semantic faithfulness measure, were applied to assess the generated responses. The results were exported as a JSON file for further analysis.

Meanwhile, for BPMN XML the generated answers are evaluated using the same metrics, with the retrieval process – explained in detail in Section 3.3 – providing the necessary context for each query. Just as in the RDF case, these outputs are compiled into a dataset and exported as a JSON file containing the user input, retrieved contexts, response, reference and the four metrics.

Within our experiments, it is crucial to evaluate the overall accuracy of the generated answers as complete units rather than disjointed facts. Therefore, we configured the factual correctness metric with the following parameters and values:

```
mode = “precision”, atomicity = “low”, coverage = “high”
```

We set the mode to *precision*, because we want our evaluation to strictly reward accuracy, ensuring that every detail in the generated answer is correct. While the default mode balances *precision* and *recall* and the *recall* mode might accept a more complete answer even if some details are off, *precision* mode penalizes any incorrect or unnecessary information. This approach is essential for our experiments, where even small factual errors can lead to misleading interpretations of the analyzed processes. Moreover, since BPMN elements are highly interconnected, we found that setting atomicity to *low* is more suitable – given that the answer and ground truth are fragmented into fewer claims – compared to a *high* atomicity setting, which would break each of these phrases into many small, dispersed pieces. Using *high* coverage ensures that the entire ground truth is reflected in the response, without omitting any details. In contrast, *high* atomicity and *low* coverage would have allowed for a fragmented evaluation that might overlook the holistic structure of our processes.

The XML process serialization offers a standard vocabulary, not only a standard structure, XML being traditionally a data interchange format. The RDF export of BPMN is only standard in the structural sense – as a directed semantic graph standard – and not as a process description vocabulary, for which intuitive tool-specific terms are generated by Bee-Up, derived from the concept labeling that is prescribed on the metamodel level. Therefore, LLMs may treat the RDF variant as a semantic network that happens to contain edges whose labels describe workflows, while the XML variant can be recognized upfront as a standard interchange format that is widely available in training corpora and Web content.

We selected test queries to span a range of retrieval cases of different complexities: from simple element enumeration/type distinctions to navigating conditional branching and inter-pool communication, collecting data annotations (e.g. times, costs) and for the realistic example, also the navigation of the subprocess hyperlink.

The strategy was rooted in a perspective based on our metamodeling experience, which is common to both the XML and RDF serializations – in the sense that they explicitly distinguish node and relation types, containment and data annotation; human interpretation of any BPMN diagrams cognitively relies on grasping these distinctions and, based on them, navigating multi-hop relationships within directed graphs. Experiments are also guided by the BPMN usage trends reported by [38], where the most common constructs found in several large collections of BPMN diagrams were identified – our battery of examples does not yet cover all possible constructs, as we are prioritizing based on such usage rankings.

We also diversified prompts according to the business process management perspectives – i.e. looking at control flows, communication/collaboration perspective and the data perspective, as well as subprocess navigation between different diagrams as expected in a realistic process repository where not all information is expressed by a single diagram. To avoid erroneous interpretations, we tried to formulate the queries as explicitly as possible, consistently using standard BPMN terminology - for instance, referring to process steps as tasks.

Our findings are presented in the following tables, with the highest metric value highlighted in bold, accompanied by a detailed discussion. Occasional explanations of how the metrics are calculated will be provided for certain scenarios; in some cases, the scores are sufficiently self-explanatory. Being LLMs are probabilistic by design, multiple test runs verified the most consistently retrieved responses, regardless of variations in phrasing or style.

Sections 4.1 and 4.2 will discuss the most relevant prompts where meaningful differences can be highlighted, whereas less interesting prompts, where an inability to respond or less significant differences manifested, can be consulted in the appendix.

4.1. Prompts on the explicit realistic example

Table 1. Comparative analysis of GPT-4's responses on task and subprocess enumeration in the "User coordinated bot (at Client)" pool (Figure 4).

Prompt	**Enumerate all tasks and all subprocesses inside 'User coordinated bot' pool, subsequent to the task labeled 'Add product to cart'.**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **OpenAI's GPT-4 model via API on BPMN XML formatted data**)
Response	The tasks and subprocesses inside 'User coordinated bot' pool, subsequent to the task labeled 'Add product to cart' are the following: 'Edit quantity', 'Bot authentication microflow', 'Authenticate user of failed authentication', 'Provide delivery & invoicing data', 'Request billing', 'Trigger payment with saved payment method', 'Approve online payment', 'Look for email from merchant', 'Save invoice', 'Generate order confirmation SMS', 'Generate failure SMS', 'Track order', 'Notify delay', and 'Leave feedback'.	The tasks and subprocesses from the 'Robot in browser' lane that are subsequent to the task labeled 'Add product to cart' are 'Edit quantity', 'Provide delivery & invoicing data', 'Request billing', 'Trigger payment with saved payment method'.
Ground truth	All tasks and all subprocesses inside 'User coordinated bot' pool, subsequent to the task labeled 'Add product to cart' are: 'Edit quantity' (belonging to the subprocess labeled 'Repeat for each desired product'), 'Bot authentication microflow' (subprocess), 'Provide delivery & invoicing data', 'Notify user of failed authentication', 'Request billing', 'Trigger payment with saved payment method', 'Approve online payment', 'Look for email from merchant', 'Save invoice', 'Generate order confirmation SMS', 'Generate failure SMS', 'Track order', 'Leave feedback', 'Notify delay'.	
Response relevancy	**0.94**	0.81
Factual correctness	**0.93**	0.12

Semantic similarity	**0.99**	0.86
Semantic faithfulness	**0.82**	0.76

Looking at the results in Table 1, Case I (RDF formatted data) and Case II (BPMN XML export) behave rather differently. In the first case, where we filter the RDF triples with the SPARQL query, GPT-4 provides the user with an almost complete and correct answer that also closely matches the ground truth – evidenced by high relevancy, correctness and semantic similarity scores. The slightly lower score of 0.82 for semantic faithfulness indicates that most of the important details from the structured context are properly reflected within the generated response, except for the expanded subprocess labeled "Repeat for each desired product", where the tasks "Add product to cart" and "Edit quantity" are displayed.

In contrast, the second case yields a partial response, enumerating only the first few tasks following the task labeled "Add product to cart" within the requested pool. Given that the BPMN XML file exceeds 20000 tokens and GPT-4's context window is limited we opted for *RecursiveCharacterTextSplitter*[11] to automatically split the text into chunks of 5000 characters with an 800-character overlap, ensuring that contextual information at the edges of each chunk is not lost during the splitting process. By default, our retriever returns 4 chunks per query, which falls within a safe limit for the input. Nonetheless, for lengthy process descriptions, where tasks and subprocesses are distributed throughout, this default setting sometimes leads to only the first 4 chunks being retrieved, which explains the partial response (see Table 1, Case II). It is, however, possible to combine manual tag removal with automated text splitting, or to increase the number of retrieved chunks to capture more context.

Another observation about Case II is that GPT-4 failed to capture the subprocesses embedded within the BPMN model, focusing solely on task-type elements and contributing to an incomplete representation even of this small portion of the process that the LLM retrieved.

Table 2. Comparative analysis of GPT-4's responses on simultaneous task execution (Figure 4).

Prompt	**Can the tasks 'Notify user of failed authentication' and 'Provide delivery & invoicing data' be executed simultaneously and why or why not?**	
Criteria	Case I	Case II
Response	Based on the model, the tasks 'Notify user of failed authentication' and 'Provide delivery & invoicing data' cannot be executed simultaneously. This is because these tasks are conditioned by different outcomes of the 'Authentication_successful?' gateway. The task 'Notify user of failed authentication' is triggered when authentication is unsuccessful, whereas the task 'Provide delivery & invoicing data' is triggered when authentication is successful. Thus, these tasks target opposing scenarios and cannot occur at the same time.	The tasks 'Notify user of failed authentication' and 'Provide delivery & invoicing data' cannot be executed simultaneously. This is because in the given context, these tasks are part of a sequence flow in a business process model, which means they are executed one after the other, not concurrently. The 'Notify user of failed authentication' task is executed when the authentication is unsuccessful, while the 'Provide delivery & invoicing data' task is executed after successful authentication.
Ground truth	No, the tasks 'Notify user of failed authentication' and 'Provide delivery & invoicing data' cannot be executed simultaneously because they follow an exclusive gateway labeled 'Authentication successful?', allowing only one path to be executed. The process either handles a failed authentication or proceeds with a successful authentication, but not both.	

---

[11] https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/recursive_text_splitter/

Response relevancy	**0.99**	**0.99**
Factual correctness	**1.00**	0.25
Semantic similarity	**0.97**	0.96
Semantic faithfulness	**0.79**	0.78

Both responses in Table 2 score very high on relevancy (0.99) and semantic similarity (0.97 for Case I and 0.96 for Case II), which means they both address the prompt very well and capture the overall meaning expected by the ground truth, respectively. However, concerning factual correctness, Case I scores a perfect 1.00, while Case II only scores 0.25. This discrepancy shows that although the XML-based answer (Case II) sounds similar in meaning, it misses or misrepresents key factual details – specifically, it fails to fully capture the exclusive gateway mechanism, which is the true reason that prevents simultaneous execution and not the fact that they are executed one after another. Semantic faithfulness is calculated by comparing the vector embeddings of the generated answer and the retrieved context. In our scenario, a score around 0.79 for Case I shows that most key details are captured, even though minor discrepancies exist. For example, in the RDF context, subjects and objects are expanded to full IRIs using the default namespace defined by the prefix ":" (which, in this case, is "http://www.omilab.org#"). This means that a snippet like:

```
<http://www.omilab.org#Task_BPMN-19347-Provide_delivery_+_invoicing_data> a
<http://bee-up.omilab.org/rdf/1_7#o_Task_BPMN>
```

can slightly alter the vector representation compared to the generated answer, leading to small mismatches in the cosine similarity. Equivalently, Case II's score of 0.78 reflects comparable alignment with its context, despite the differences in how the information is structured.

Table 3. Comparative analysis of GPT-4's responses on "Bot authentication microflow" (subprocess) description under a condition (Figure 4).

Prompt	**What happens inside the corresponding subprocess if the user credentials are not valid?**	
Criteria	Case I	Case II
Response	If the user credentials are not valid, the subprocess in this context diverges at the Exclusive Gateway "Credentials valid?". It will then execute the Task "Check for multiple failed attempts", followed by another decision at "Attempts under limit?". If the number of attempts is under the limit a grace period would be given for another attempt otherwise, the "Lock account" task runs, leading to "Increment failed attempts counter", and "Notify user of account status" tasks. A notification is then sent out, marking the end of the flow with the End Event "Failed authentication".	If the user credentials are not valid, the process diverges at the "Credentials valid?" gateway. The process then moves to the "Notify user of failed authentication" task, where the user is informed about the failed authentication attempt.
Ground truth	If the user credentials are not valid, the system checks for multiple failed attempts. If there are no more attempts, the account is locked, the failed attempts counter is incremented, the user is notified of the account status and the subprocess ends with a failed authentication. If the attempts are under limit, the user is granted a grace period for one more try before lockout and if the user accepts the additional authentication attempt, the system returns to the earlier task of retrieving user credentials.	

Response relevancy	**0.91**	**0.91**
Factual correctness	**0.87**	0.33
Semantic similarity	**0.93**	0.89
Semantic faithfulness	0.77	**0.78**

The part "inside the corresponding subprocess" from this prompt (Table 3) is designed to test whether the model can identify the correct subprocess – namely, "Bot authentication microflow" – based on a clear condition, without being explicitly told. Both responses do recognize it by mentioning the exclusive gateway, "Credentials valid?", which represents the central divergence inside the subprocess. However, while the RDF-based answer in Case I accurately goes further by elaborating upon tasks such as checking for multiple failed attempts, granting a grace period and locking the account, the XML-based answer in Case II stops prematurely at "Notify user of failed authentication" task, which is not even part of the subprocess in question. This omission leads to a much lower factual correctness score (0.33) for Case II.

Case II's high semantic similarity score (0.89) may seem surprising given its factual incompleteness, but it can be explained by how the metric is calculated. Semantic similarity is determined by comparing vector embeddings of the generated answer and the ground truth, focusing on the overall meaning rather than every fine-grained detail. OpenAI [32,33] explains that embeddings convert text into dense vector representations that capture the semantic essence of the input, enabling tasks like semantic search and similarity comparisons. As mentioned before, we use OpenAI's *text-embedding-ada-002* engine, which is specifically designed to produce embeddings that reflect the overall, general meaning of the text. As a result, even if the answer omitted some parts, its main semantic content remains closely aligned with the ground truth – capturing the core idea that the process diverges at the "Credentials valid?" gateway and can also follow a failure branch – thus yielding a high similarity score.

Table 4. Comparative analysis of GPT-4's responses on cost and execution time of the tasks within a subprocess (Figure 4).

Prompt	**What are the cost and execution time for every task within the bot authentication microflow?**	
Criteria	Case I	Case II
Response	The cost and execution time for each task within the bot authentication microflow are as follows: 'Retrieve user credentials' costs 0.01 and takes 2 seconds, 'Validate credentials' costs 0.02 and takes 5 seconds, 'Check access rights' costs 0.03 and takes 6 seconds, 'Create authentication log' costs 0.01 and takes 1 second, 'Check for multiple failed attempts' costs 0.01 and takes 1 second, 'Lock account' costs 0.03 and takes 5 seconds, 'Increment failed attempts counter' costs 0.01 and takes 1 second, and 'Notify user of account status' costs 0.02 and takes 2 seconds.	The bot authentication microflow consists of several tasks, each with its own cost and execution time. The "Activate bot" task has an execution cost of 0.05 and takes 00:00:05 to execute. The "Check for multiple failed attempts" task has an execution cost of 0.01 and takes 00:00:03 to execute. The "Lock account" task does not have specified execution costs and time in the provided context. The "Check access rights" task has an execution cost of 0.03 and takes 00:00:06 to execute. The "Create authentication log" task has an execution cost of 0.01 and takes 00:00:01 to execute.
Ground truth	The cost and execution time for every task within the bot authentication microflow are: a cost of 0.01 and execution time of 2 seconds for 'Retrieve user credentials', a cost of 0.02 and execution time of 5 seconds for 'Validate credentials', a cost of 0.03 and execution time of 6 seconds for 'Check access rights' a cost of 0.01 and execution time of 1 second for 'Create authentication log', a cost of 0.01 and execution time of 1 second for 'Check for multiple failed attempts', a cost of 0.03 and execution time of 5 seconds for 'Lock	

	account', a cost of 0.01 and execution time of 1 second for 'Increment failed attempts counter', a cost of 0.02 and execution time of 2 seconds for 'Notify user of account status'.	
Response relevancy	**0.99**	**0.99**
Factual correctness	**1.00**	0.60
Semantic similarity	**0.99**	0.96
Semantic faithfulness	**0.80**	0.77

Table 4 illustrates a specific example where the GPT model is challenged to extract certain process performance metrics from the BPMN process. Both responses score very high in relevancy (0.99) and semantic similarity (0.99 for Case I and 0.96 for Case II), showing that both answers grasp the core idea: detailing the cost and execution time within "Bot authentication microflow". However, Case I achieves perfect factual correctness (1.00) by accurately listing all eight tasks as specified in the ground truth. In contrast, Case II only correctly identifies four tasks and even mistakenly includes the "Activate bot" task, which does not belong to the subprocess. This difference arises because the XML standard uses a closed-world schema, whereas the RDF export offers a more flexible and semantically enriched network of associations. The semantic faithfulness scores (0.80 for Case I and 0.77 for Case II) indicate that both methods capture most of the essential context, while the XML-based approach's rigidity results in a lower factual correctness score (0.60) and only a partial capture of the subprocess details.

4.2. Prompts on the Non-Explicit Minimalist Models

Building on the experiments outlined in Section 4.1, we now shift our focus to exploring minimalist, non-explicit patterns. Using the same setup and evaluation metrics, we examine the patterns illustrated in Figures 5 and 6 to uncover how these streamlined models – represented in RDF and BPMN XML – perform under a diversity of queries that include various perspectives, such as temporal understanding, participant involvement, interactions and associations.

Table 5. Comparative analysis of GPT-4's responses on simple task sequence identification (Figure 5-a).

Prompt	**Identify the tasks in the order described in the process.**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The tasks in the order described in the process are A, B, and C.	The tasks in the order described by the process are Task A, Task B, and Task C.
Ground truth	The tasks in the order described in the process are: A, B, C.	
Response relevancy	**0.97**	**0.97**
Factual correctness	**1.00**	**1.00**
Semantic similarity	**0.99**	0.98
Semantic faithfulness	**0.78**	0.74

The prompt “Identify the tasks in the order described in the process” from Table 5 is straightforward, asking the GPT model to sequentially list the tasks from the BPMN model represented in Figure 5-a. Both responses correctly identify the tasks A, B and C, which accurately match the ground truth, resulting in perfect factual correctness (1.00) in both cases. The semantic similarity scores are also very high – 0.99 for Case I and 0.98 for Case II – in spite of minor wording differencesThe semantic faithfulness scores (0.78 for Case I and 0.74 for Case II) indicate that the essential context from this pattern is well captured; the slight drop in Case II is due to the additional verbosity from the prefix “Task”, which slightly alters the vector representation, but without impacting the core semantic content.

We also tested additional prompts on models from Figures 5-b, 5-e and 6-l, to assess whether GPT-4 can accurately handle temporal sequencing (as shown in Table 9), recognize branches (Table 10) and spatial-temporal queries (Table 11). Since the responses and the metric scores were correct for both data formats in all three scenarios – similar to the metric scores from Table 5 – we have grouped these findings in the Appendix section.

Table 6. Comparative analysis of GPT-4’s responses on event identification and their trigger in a taskless sequence, without exemplifying possible trigger types (Figure 5-c).

Prompt	**Identify all the events before the event Z, in the order executed in the process, and their trigger.**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI’s GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The events before event Z are, in the order executed in the process: “Start Event (BPMN)-35353”, “X”, and “Y”. The “Start Event (BPMN)-35353” is initiated by the start of the process. Event “X” is triggered by a message, as well as event “Y”.	The events before event Z are event Y and event X. Event Y is triggered by a message event definition with id “sid-8dd832f4-e8e5-4878-b075-5012f1dfa9af”. Event X is also triggered by a message event definition with id “sid-7699c4c7-51aa-4401-813e-96d2a9cc4835”.
Ground truth	The events before the event Z, in the order executed in the process, are: Start event, X and Y. The trigger for Start is not specified. The trigger for X is a message. The trigger for Y is also a message.	
Response relevancy	**0.93**	0.92
Factual correctness	**0.40**	0.33
Semantic similarity	**0.94**	**0.94**
Semantic faithfulness	**0.79**	0.72

Overall, the generated answers in both cases (Table 6) convey the same core meaning as the ground truth, given the high semantic similarity scores (0.94). However, the factual correctness is notably lower – 0.40 for Case I and 0.33 for Case II – indicating that, while the main idea is captured, some key details are either missed, misrepresented or both. In Case I, the answer correctly identifies the start event along with the events X and Y, both triggered by messages (see Figure 5-c), but it also includes an extra detail about the trigger of the start event that is not specified in the ground truth. Moreover, since the start event is not explicitly labeled in the context like the rest of the events, GPT-4 defaults to using the full generic label assigned by the modeling tool, which differs from what we call it in the ground truth (i.e., just “Start event”).

In contrast, Case II's XML-based response entirely disregards the start event. It also suggests that one task triggers the next. Although this interpretation makes sense to some extent, it certainly does not match what was expected in the ground truth/reference, leading to an even more significant drop in factual correctness. The semantic faithfulness scores (0.79 for Case I and 0.72 for Case II) further confirm that and even though most of the essential context is retrieved, the XML-based answer loses some important details, resulting in a lower overall match with the ground truth.

Table 7. Comparative analysis of GPT-4's responses on event identification and their trigger in a taskless sequence, exemplifying possible trigger types (Figure 5-c).

Prompt	**Identify all the events before the event Z, in the order executed in the process, and their trigger (Cancel, Compensation, Conditional, Error, Escalation, Link, Message, Signal, Timer).**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The events before the event Z are the "Start Event (BPMN)-35353", event X, and event Y. The Start Event (BPMN)-35353 does not have any particular triggers, event X is triggered by a message, and event Y is also triggered by a message.	The events before event Z are event X and event Y. Event X is an intermediate throw event triggered by a message as indicated by the "messageEventDefinition" tag. Event Y is an intermediate catch event also triggered by a message as indicated by the "messageEventDefinition" tag.
Ground truth	The events before the event Z, in the order executed in the process, are: Start event, X and Y. The trigger for Start is not specified. The trigger for X is a message. The trigger for Y is also a message.	
Response relevancy	**0.89**	**0.89**
Factual correctness	**0.50**	0.43
Semantic similarity	**0.94**	0.92
Semantic faithfulness	**0.85**	0.75

We tested on the same BPMN model (in Figure 5-c) with another prompt to see if a hint would boost the quality of the generated answers. In the initial prompt (see Table 6), we simply asked for the events before event Z and their triggers, while in the second (see Table 7), we provided a list of possible trigger types (Cancel, Compensation, Conditional, Error, Escalation, Link, Message, Signal, Timer) as a hint. Case I scored high in both scenarios (Tables 6 and 7) regarding relevancy and semantic similarity, which means that even without the hint, the core idea was captured accurately. However, the hint in the second prompt led to marked improvements in factual correctness and semantic faithfulness. For both cases factual correctness and semantic faithfulness improved due to the hint. These enhancements suggest that overall, the additional hint helped GPT-4 to provide more detail, reducing ambiguity in what constitutes a trigger. According to RAGAs, the relevancy metric is computed by reverse-engineering several question variants from the generated answer and then measuring how closely these reconstructed questions match the original question/prompt using cosine similarity. In our example, the additional details (all possible trigger types) introduced in the original question could be potential factors that contribute to a slight decrease of metric value, from 0.93 to 0.89.

Table 8. Comparative analysis of GPT-4's responses on boundary branching effect in the process (Figure 5-g).

Prompt	**How does the boundary message event m affect the process?**	
Criteria	CASE I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	CASE II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The boundary message event "m" in this process acts as an interrupting event. It disrupts the normal flow of the process and directs it towards another process step. In this case, after the event "m" happens, the process proceeds to Task D instead of continuing its normal sequence.	I do not know.
Ground truth	If the boundary message event m arrives during the execution of the task B, this task is interrupted and task D will be executed instead. After task D is completed, the process ends at the subsequent End event.	
Response relevancy	**0.95**	0.00
Factual correctness	**1.00**	0.00
Semantic similarity	**0.94**	0.70
Semantic faithfulness	**0.84**	0.71

In our experiment from Table 8, even though the RAG pipeline successfully retrieved the XML context with the complete process flow (since the process was not lengthy), GPT-4 still responded with "I do not know." when asked about the effect of the boundary message event. This outcome aligns with the instructions provided in the prompt template, namely "If you do not know the answer to the following question, just say 'I do not know.' or 'The provided context does not contain enough information.' and nothing else.". XML's verbose, deeply nested structure, as well as its lack of explicit semantic relationships between elements, hinder GPT-4 from extracting and interpreting these nuanced process details. This contributes to the XML-based response scoring zero in both relevancy and factual correctness. Despite the XML-based response being simply "I do not know.", its semantic similarity score remains around 0.70. Research [37] has shown that even minimal expressions can carry rich semantic signals, meaning that, even though "I do not know" contains almost no factual details, its vector embedding is prone to capturing a generic signal of uncertainty, as embedding models like *text-embedding-ada-002* are designed to "behave". In our case, the ground truth describes a specific conditional process (interruption at task B leading to task D), which inherently involves a notion of deviation or uncertainty about the normal flow, thus falling within a similar semantic domain as the generated answer's. This shared aspect of uncertainty, even though very broadly defined, could be the cause of a moderate similarity score of 0.70, rather than something much lower, as one would expect. Similarly, semantic faithfulness is computed using cosine similarity between the embeddings of the retrieved context and the generated answer, which means that even a minimal response can yield a moderate score if it captures enough high-level semantic features. Therefore, in similar scenarios, any response with semantic similarity and faithfulness scores below 0.75 can be considered unsatisfactory.

This phenomenon is also evident in our other, more challenging text queries – those that extend beyond simple task identification or basic sequence flows. In our parallel branching tests (Table 12), for example, we asked whether a certain task could be executed if a preceding task, from a parallel branch, had not been already executed and why that might be the case. Similarly, our event branching test (see Table 13) examined if two tasks could be executed concurrently, even if they originated from two separate branches emerging from a XOR event gateway – where only one path is typically taken. The query referring to the compensation branching model (see Table 14) evidenced in Figure 5-h, challenged GPT-4 to correctly identify the cause for the compensation flow, while our task collaboration tests (Tables 15 and 16) examined whether the model could accurately depict interactions among participants in the process (marked by Message Flows in the BPMN standard). Across these 5 scenarios, we observe a similar pattern in the metrics indicating that, with the increased complexity and nuanced BPMN logic in such non-explicit models, the RDF-based approach always outperforms the XML-based one.

Our final experiments on BPMN models involving event collaboration (Figure 6-k) and lane coordination with data (Figure 6-m) further solidify the idea that RDF provides a more detailed and semantically rich representation of BPMN models than XML. However, these experiments also produced some rather intriguing scores that warrant some discussion.

Firstly, it is worth mentioning how Case I and Case II, in both scenarios (see Tables 17 and 18, respectively), achieved a perfect relevancy score (1.00), even if Case II, unlike Case I, is only partially correct. As previously mentioned, this high relevancy can be caused by the way the metric is calculated (explained previously). Looking at the results from Table 17, the generated answer for Case II only mentions the event "E2" and does not explicitly reference the message "M1". Keeping this in mind, the generated questions from the generated answer are inclined to align closely with the original question – specifically, "What does the start of the process executed by P2 depend on?" – which also does not directly mention M1; we believe that this might impact the relevancy score, despite the factual details being incomplete. This behavior was also met in a previous example (see Table 5-c).

Secondly, the low to medium factual correctness scores, despite Case I giving fully correct answers in both scenarios, might stem from the evaluation metric's sensitivity to the granularity of the generated claims. For instance, GPT-4 provided additional details from the retrieved context – such as the "Subsequent" relationship from task B to task C or the "Data Association" linking a data object to a task (see Table 18, Case I) – that are not explicitly required by the ground truth.

5. Discussion and Implications

In every prompt scenario (Table 1 – Table 18), the RDF-based representation (Case I) exhibits superior performance across almost all evaluation metrics. Built on explicit subject–predicate–object triples that map out clear, interconnected relationships between BPMN elements, the RDF-represented processes aid the OpenAI's GPT-4 model in providing complete and accurate responses with high scores in relevancy, factual correctness, semantic similarity and semantic faithfulness. In contrast, the BPMN XML-based response was occasionally deficient, particularly in factual correctness. These behaviors are observed in prompts requiring the extraction of temporal and conditional process details, where RDF consistently captures the nuanced relationships inherent in BPMN models more effectively than BPMN XML (with a more rigid format). These results underscore the importance of employing RDF as a means to serialize BPMN processes, thereby reinforcing the notion that structured, semantically rich representations can enhance natural language process querying tasks. Table 19 highlights which case achieved higher scores for each metric across all evaluation scenarios.

Table 19. Evaluation synthesis

Metric	Overall comparison	Notable examples
Response relevancy	**Case I** generally scores higher compared to **Case II**.	• Task and subprocess sequence inside a pool (Table 1); • Branching scenarios: exclusive (Table 2), boundary (Table 8), parallel (Table 12), event (Table 13), compensation (Table 14);

		- Task and collapsed pool collaboration (Tables 15 and 16, respectively); - Lane coordination with data (Table 18).
Factual correctness	**Case I** consistently outperforms **Case** II, many times even by a significant margin.	- Task and subprocess sequence inside a pool (Table 1); - Task sequence inside a subprocess based on a condition (Table 3); - Branching scenarios: exclusive (Table 2), boundary (Table 8), parallel (Table 12), event (Table 13), compensation (Table 14); - Task, collapsed pool and event collaboration (Tables 15, 16 and 17, respectively);
Semantic similarity	**Case I** exhibits higher similarity scores across all scenarios.	- All scenarios, including lane coordination with data processes (Table 18).
Semantic faithfulness	**Case I** is generally higher, although, in one instance (Table 3), the scores are nearly equal (**Case I = 0.77** vs. **Case II = 0.78**)	- All scenarios, except for task sequence inside a subprocess based on a condition (Table 3).

Broader implications of these findings extend beyond the immediate context of BPMN process analysis. Enterprises may benefit from migrating toward semantically enriched process representations, where the main purpose is not to have the process description as a closed world data structure to be leveraged only by automation or simulation engines, but as a knowledge graph that adds on top of the basic machine-readability requirement the possibility of semantic enrichment, interpretation and integration across future knowledge-based systems that leverage procedural knowledge available through established notations such as BPMN. Both tool vendors and standardization bodies may want to consider a shift from the XML-dominated world of process interchange and serialization. Current implementations are only available as experimental artifacts, typically developed under a Design Science umbrella,

Further implications that propagate across the full BPM lifecycle must consider what possibilities are enabled by this shift – i.e. for semantics-driven enactment, monitoring, discovery and so on, with a potential for agile domain-specificity to be added as a semantic layer over the core process or trace descriptions.

6. Conclusions

Through the comparative use of RDF-encoded semantic graphs and XML-encoded diagrams, the experiments delineate nuanced treatments for BPMN models subjected to OpenAI's GPT-4 as procedural knowledge to be interpreted. Our findings, substantiated by the scores obtained using the RAGAs framework, indicate that RDF exports provide a more open-ended and relationship-aware approach to process interpretation, compared to the standard XML export which appears to be treated as a closed-world data structure, even though the RDF version employs a non-standard, tool-specific, process description vocabulary.

The study has inherent limitations due to the fast evolution and stochastic nature of LLM services – not only variations between different versions, but also between work sessions are noticeable. Therefore, this work is not intended to be an evaluation of capabilities of a certain product, but more of a proposition of an interpretation and analysis protocol towards an augmented version of the BPM lifecycle as proposed by [10]. We are also reluctant in stating that these are experiments on "process understanding" due to the inherent limitations of what tokenized language models "understand" as reported by recent investigations [39]; consequently we are treating process descriptions as information structures that can be queried and navigated along explicit references by the pattern matching mechanisms of LLMs, hence our current focus on serializations instead of computer vision or conversational modeling.

Future work will be invested in further exploration of the possible synergies between the BPM lifecycle and LLM services, specifically on prompting strategies that can generate such process serializations besides interpreting them. We also see opportunities involving domain-specific knowledge graphs, meta-models and XML/RDF schemas that could yield process interpretations which are better contextualized in relation to the complexity of the prompt and the knowledge graph patterns that can act as process context. Pre-training and fine-tuning of public GPT services based on domain-specific knowledge is another direction to be explored towards a hybridization of model-driven engineering and prompt engineering in service of enterprise-tailored business process management.

References

[1] Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of business process management* (2nd ed.). Springer Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-662-56509-4.

[2] Camunda (n.d.), What is BPMN? Business Process Model and Notation. https://camunda.com/bpmn/.

[3] Buchmann, R., Eder, J., Fill, H. G., Frank, U., Karagiannis, D., Laurenzi, E., Mylopoulos, J., Plexousakis, D., Santos, M. Y. (2024): Large language models: Expectations for semantics-driven systems engineering, Data & Knowledge Engineering 152, article 10234, doi: https://doi.org/10.1016/j.datak.2024.102324.

[4] OMILAB NPO (2024), Bee-Up For Education. https://bee-up.omilab.org/activities/bee-up/.

[5] Bachhofner, S., Kiesling, E., Revoredo, K., Waibel, P. and Polleres, A. (2022). Automated Process Knowledge Graph Construction from BPMN Models. Database and Expert Systems Applications. 13426, pp. 32-47. Springer, Cham. doi:http://dx.doi.org/10.1007/978-3-031-12423-5_3.

[6] Uifălean, Ș., Ghiran, A. and Buchmann, R. (2023). Employing Graph Databases for Business Process Management and Representation. Advances in Information Systems Development (pp. 73-92). Springer, Cham. doi:http://dx.doi.org/10.1007/978-3-031-32418-5_5.

[7] Fill, H., Fettke, P. and Köpke, J. (2023). Conceptual Modeling and Large Language Models: Impressions From First Experiments With ChatGPT. Enterprise Modelling and Information Systems Architectures (EMISAJ), 18, 1-15. doi:https://doi.org/10.18417/emisa.18.3.

[8] Dolha, D., Buchmann, R. A. (2024). Generative AI for BPMN Process Analysis: Experiments with Multi-modal Process Representations. In Proceedings of BIR 2024, LNBIP 529, pp. 19-35, Springer, https://doi.org/10.1007/978-3-031-71333-0_2.

[9] Guntur, B. H. (2024). Automating Data Flow Diagram Generation from User Stories Using Large Language Models. 7th Workshop on Natural Language Processing for Requirements Engineering. Retrieved from https://hal.science/hal-04525925/.

[10] Vidgof, M., Bachhofner, S. and Mendling, J. (2023). Large Language Models for Business Process Management: Opportunities and Challenges. doi:https://doi.org/10.48550/arXiv.2307.09923.

[11] BOC GmbH. (2024). ADOxx AQL query language. https://www.adoxx.org/live/adoxx-query-language-aql.

[12] Grohs, M., Abb, L., Elsayed, N. and Rehse, J. (2023). Large Language Models can accomplish Business Process Management Tasks. Business Process Management Workshops. BPM 2023. Lecture Notes in Business Information Processing. 492, pp. 453-465. Springer, Cham. doi:https://doi.org/10.1007/978-3-031-50974-2_34.

[13] Jalali, A. (2021). Graph-Based Process Mining. International Conference on Process Mining (pp. 273-285). Springer, Cham. doi:https://doi.org/10.1007/978-3-030-72693-5_21.

[14] OpenAI (2023). GPT-4. https://openai.com/research/gpt-4.

[15] Bachhofner, S., Kiesling, E., Revoredo, K., Waibel, P. and Polleres, A. (2022). Automated Process Knowledge Graph Construction from BPMN Models. Database and Expert Systems Applications. 13426, pp. 32-47. Springer, Cham. doi:http://dx.doi.org/10.1007/978-3-031-12423-5_3.

[16] Polyvyanyy, A. (2021). Process Querying: Methods, Techniques, and Applications. Process Querying Methods (pp. 511-524). Springer, Cham. doi:https://doi.org/10.1007/978-3-030-92875-9_18.

[17] Kourani, H., Berti, A., Schuster, D., van der Aalst, W.M.P. (2024). Process Modeling with Large Language Models. In: van der Aa, H., Bork, D., Schmidt, R., Sturm, A. (eds) Enterprise, Business-Process and Information Systems Modeling.

BPMDS EMMSAD 2024 2024. Lecture Notes in Business Information Processing, vol 511. Springer, Cham. doi:https://doi.org/10.1007/978-3-031-61007-3_18.

[18] Busch, K. R. (2023). Just Tell Me: Prompt Engineering in Business Process Management. Enterprise, Business-Process and Information Systems Modeling (BPMDS EMMSAD 2023). Lecture Notes in Business Information Processing. 479, pp. 3-11. Springer, Cham. doi:https://doi.org/10.1007/978-3-031-34241-7_1.

[19] Ayad, S. and Alsayoud, F. (2024). Prompt engineering techniques for semantic enhancement in business process models. Business Process Management Journal, 30(7), 2611-2641. doi:https://doi.org/10.1108/bpmj-02-2024-0108.

[20] Gutschmidt, A., Nast, B. (2025). Assessing Model Quality Using Large Language Models. In: Paja, E., Zdravkovic, J., Kavakli, E., Stirna, J. (eds) The Practice of Enterprise Modeling. PoEM 2024. Lecture Notes in Business Information Processing, vol 538. Springer, Cham. doi:https://doi.org/10.1007/978-3-031-77908-4_7.

[21] Jasińska, K., Lewicz, M. and Rostalski, M. (2023). Digitization of the enterprise - prospects for process automation with using RPA and GPT integration. 27th International Conference on Knowledge Based and Intelligent Information and Engineering Sytems. 225, pp. 3243-3254. Procedia Computer Science. doi:https://doi.org/10.1016/j.procs.2023.10.318.

[22] Shahin, M., Chen, F., Hosseinzadeh, A., Maghanaki, M. and Eghbalian, A. (2024). A Novel Approach to Voice of Customer Extraction using GPT-3.5 Turbo: Linking Advanced NLP and Lean Six Sigma 4.0. (A. Nee, K. Cheng, D. Russell, M. Shunmugam and I. Lazoğlu, Eds.) The International Journal of Advanced Manufacturing Technology, 131, 3615-3630. doi:http://dx.doi.org/10.21203/rs.3.rs-3246823/v1.

[23] Yang, L., Chen, H., Li, Z., Ding, X. and Wu, X. (2024). Give Us the Facts: Enhancing Large Language Models with Knowledge Graphs for Fact-aware Language Modeling. IEEE Transactions on Knowledge and Data Engineering, 36, 3091-3110. Retrieved from https://api.semanticscholar.org/CorpusID:259203671.

[24] Karagiannis, D., Buchmann, R. A. and Utz, W. (2022). The OMiLAB Digital Innovation environment: Agile conceptual models to bridge business value with Digital and Physical Twins for Product-Service Systems development. Computers in Industry, 138. doi:https://doi.org/10.1016/j.compind.2022.103631.

[25] SAP (2024), SAP Signavio Process Transformation Suite, Academic Edition. Retrieved from SAP Signavio: https://www.signavio.com/academic-and-research-alliances/.

[26] Buchmann, R. A. and Karagiannis, D. (2015). Pattern-based Transformation of Diagrammatic Conceptual Models for Semantic Enrichment in the Web of Data. Knowledge-Based and Intelligent Information & Engineering Systems 19th Annual Conference (KES-2015). 60, pp. 150-159. Procedia Computer Science. doi:http://dx.doi.org/10.1016/j.procs.2015.08.114.

[27] Ontotext. (2025). What is GraphDB?: https://graphdb.ontotext.com/documentation/10.8/index.html.

[28] Iga, V.I.R., Silaghi, G.C. (2024). LLMs for Knowledge-Graphs Enhanced Task-Oriented Dialogue Systems: Challenges and Opportunities. In: Almeida, J.P.A., Di Ciccio, C., Kalloniatis, C. (eds) Advanced Information Systems Engineering Workshops. CAiSE 2024. Lecture Notes in Business Information Processing, vol 521. Springer, Cham. https://doi.org/10.1007/978-3-031-61003-5_15

[29] Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., Li, Y., Gupta, A., Han, H., Schulhoff, S., Dulepet, P.S., Vidyadhara, S., Ki, D., Agrawal, S., Pham, C., Kroiz, G.C., Li, F., Tao, H., Srivastava, A., Costa, H.D., Gupta, S., Rogers, M.L., Goncearenco, I., Sarli, G., Galynker, I., Peskoff, D., Carpuat, M., White, J., Anadkat, S., Hoyle, A.M., Resnik, P. (2024). The Prompt Report: A Systematic Survey of Prompting Techniques. *ArXiv, abs/2406.06608*.

[30] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. ArXiv, abs/2005.11401.

[31] Es, S., James, J., Espinosa-Anke, L. and Schockaert, S. (2024). RAGAs: Automated Evaluation of Retrieval Augmented Generation. Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (pp. 150-158). Association for Computational Linguistics. Retrieved from https://aclanthology.org/2024.eacl-demo.16.

[32] OpenAI. (n.d.) Vector embeddings. https://platform.openai.com/docs/guides/embeddings.

[33] OpenAI. (2022). Introducing text and code embeddings. https://openai.com/index/introducing-text-and-code-embeddings/.

[34] Klievtsova, N., Benzin, JV., Kampik, T., Mangler, J., Rinderle-Ma, S. (2023). Conversational Process Modelling: State of the Art, Applications, and Implications in Practice. In: Di Francescomarino, C., Burattin, A., Janiesch, C., Sadiq, S. (eds) Business Process Management Forum. BPM 2023. Lecture Notes in Business Information Processing, vol 490. Springer, Cham. https://doi.org/10.1007/978-3-031-41623-1_19.

[35] Bellan, P., Dragoni, M., Ghidini, C. (2022). Extracting Business Process Entities and Relations from Text Using Pre-trained Language Models and In-Context Learning. In: Almeida, J.P.A., Karastoyanova, D., Guizzardi, G., Montali, M., Maggi, F.M., Fonseca, C.M. (eds) Enterprise Design, Operations, and Computing. EDOC 2022. Lecture Notes in Computer Science, vol 13585. Springer, Cham. https://doi.org/10.1007/978-3-031-17604-3_11.

[36] Sui, Y., Zhou, M., Zhou, Mi., Han, S., Zhang, D. (2023). Table Meets LLM: Can Large Language Models Understand Structured Table Data? A Benchmark and Empirical Study. doi:https://doi.org/10.48550/arXiv.2305.13062.

[37] Krugmann, J.O., Hartmann, J. Sentiment Analysis in the Age of Generative AI. *Cust. Need. and Solut.* **11**, 3 (2024). https://doi.org/10.1007/s40547-024-00143-4.

[38] Compagnucci, I., Corradini, F., Fornari, F., Re, B. (2023). A Study on the Usage of the BPMN Notation for Designing Process Collaboration, Choreography, and Conversation Models. Business & Information systems Engineering 66:43-66. https://doi.org/10.1007/s12599-023-00818-7.

[39] Saba W. S. (2023): Stochastic LLMs do not understand language: towards symbolic, explainable and ontologically based LLMs, Proceedings of ER 2023, LNCS 14320, Springer, pp. 3-19

[40] W3C Community Draft Report, RDF-star and SPARQL-star, https://w3c.github.io/rdf-star/cg-spec/editors_draft.html

Appendix

Additional test queries on non-explicit minimalist BPMN models:

Table 9. Comparative analysis of GPT-4's responses on task identification in a tasks/events sequence (Figure 5-b).

Prompt	**Name all the tasks after the event X.**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The tasks after the event X are Task B and Task C.	The tasks after the event X are task B and task C.
Ground truth	The tasks after the event X are B and C.	
Response relevancy	**0.96**	**0.96**
Factual correctness	**1.00**	**1.00**
Semantic similarity	**0.99**	**0.99**
Semantic faithfulness	**0.81**	0.74

Table 10. Comparative analysis of GPT-4's responses on task identification in a decision branching process (Figure 5-e).

Prompt	**Which are the tasks that are executed until the end of the process if condition v3 is true?**	
Criteria	Case I	Case II
Response	If condition v3 is true, then the tasks executed until the end of the process are "C" and "Y".	If condition v3 is true, task "C" and task "Y" are executed until the end of the process.
Ground truth	The tasks that are executed until the end of the process if condition v3 is true are C and Y.	
Response relevancy	**0.99**	**0.99**
Factual correctness	**1.00**	**1.00**
Semantic similarity	**0.98**	0.96
Semantic faithfulness	**0.79**	0.75

Table 11. Comparative analysis of GPT-4's responses on task identification and sequence in a task collaboration process (Figure 6-l).

Prompt	**Inside which lane is the task D? Which task comes before the task D?**	
Criteria	Case I	Case II
Response	The task D is inside the lane labeled "L2". The task that comes before task D is task C.	Task D is inside the lane named "L2". The task that comes before task D is task C.
Ground truth	The task D is inside the lane L2. The task that comes before the task D is the task C.	
Response relevancy	0.95	**0.96**
Factual correctness	**1.00**	**1.00**
Semantic similarity	**0.99**	0.98
Semantic faithfulness	**0.82**	0.74

Table 12. Comparative analysis of GPT-4's responses on task sequence identification in a parallel branching process (Figure 5-d).

Prompt	**Can the task Y be executed if the task A was not previously executed? Why or why not?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	No, the task Y cannot be executed if the task A was not previously executed. This is because, according to the information provided, task Y comes after a Non-exclusive Gateway (converging), and task A is one of the tasks that lead up to this gateway. This means that task A must be completed before the process can continue and reach task Y.	The provided context does not contain enough information.
Ground truth	The task Y cannot be executed if the task A was not previously executed, because the parallel non-exclusive gateway converges all the incoming sequence flows from the tasks A, B and C before executing the task Y.	
Response relevancy	**0.98**	0.00
Factual correctness	**1.00**	0.00
Semantic similarity	**0.96**	0.75
Semantic faithfulness	**0.80**	0.75

Table 13. Comparative analysis of GPT-4's responses on task sequence identification in an event branching process (Figure 5-f).

Prompt	**Can the tasks A and B be executed simultaneously? Why or why not?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	No, the tasks A and B cannot be executed simultaneously. The diagram describes a workflow where task B can only follow after the completion of an intermediate	The provided context does not contain enough information.

	event (e2), which is one of the possible next steps after an exclusive event-based gateway (e?). Task A also follows after a different intermediate event (e1), which is another possible outcome of the same gateway. Since an exclusive gateway only allows for one sequence to follow, tasks A and B cannot be executed at the same time.	
Ground truth	No, the tasks A and B cannot be executed simultaneously, because at the exclusive event-based gateway labeled ‘e?’, the decision is based on which of the succeeding intermediate event (e1, e2 or e3) occurs first and only one of the paths can be taken.	
Response relevancy	**0.96**	0.00
Factual correctness	**0.71**	0.00
Semantic similarity	**0.95**	0.74
Semantic faithfulness	**0.83**	0.75

Table 14. Comparative analysis of GPT-4’s responses on compensation flow identification in a compensation branching process (Figure 5-h).

Prompt	**What triggers the compensation flow in the current process?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI’s GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The compensation flow in the current process is triggered by the End Event labelled as “e”, which has its compensation attribute set to “Yes”.	I do not know.
Ground truth	The compensation flow in the current process is triggered by the compensation end event e.	
Response relevancy	**1.00**	0.00
Factual correctness	**0.50**	0.00
Semantic similarity	**0.96**	0.71
Semantic faithfulness	**0.79**	0.72

Table 15. Comparative analysis of GPT-4’s responses on task execution identification in a task collaboration process (Figure 6-i).

Prompt	**After the task B is executed, what conditions need to be met for task C to be executed?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI’s GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The execution of task C requires the completion of task F as depicted by the message flow M2 in this model. After the completion of task B, task F should be executed for task C to take place.	I do not know.
Ground truth	After the task B is executed, the execution of task C is conditioned by the arrival of the message M2 produced by the task F after its execution.	

Response relevancy	**0.90**	0.00
Factual correctness	**0.75**	0.00
Semantic similarity	**0.94**	0.72
Semantic faithfulness	**0.83**	0.72

Table 16. Comparative analysis of GPT-4's responses on participant interaction in a collapsed pool collaboration process (Figure 6-j).

Prompt	**How does the participant P1 interact with other participants of the process?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The participant P1 interacts with other participants through tasks and message flows. P1 initiates a message flow "M1" from task "B" to participant P2. Later, P2 sends back an interaction "M2" to task "D". Therefore, participant P1 is able to both send and receive interactions within the process.	I do not know.
Ground truth	The participant P1 interacts with other participants of the process by sending the message M1 from the task B to the participant P2 and receiving the message M2 from the participant P2 to the task D.	
Response relevancy	**0.99**	0.00
Factual correctness	**1.00**	0.00
Semantic similarity	**0.96**	0.74
Semantic faithfulness	**0.82**	0.71

Table 17. Comparative analysis of GPT-4's responses on participant interaction and dependencies in an event collaboration process (Figure 6-k).

Prompt	**What does the start of the process executed by P2 depend on?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The start of the process executed by P2 depends on the Intermediate Event E1 from the process executed by P1. This is indicated by the message flow M1 which runs from E1 to the start event E2 in P2.	The start of the process executed by P2 depends on the event E2 as indicated by the startEvent id="sid-87CD25C7-AA75-4667-AF37-755E03DC9681" with name="E2".
Ground truth	The start of the process executed by P2 depends on the arrival of message M1 sent by P1 from E1.	
Response relevancy	**1.00**	**1.00**
Factual correctness	**0.33**	0.00
Semantic similarity	**0.94**	0.90
Semantic faithfulness	0.79	**0.81**

Table 18. Comparative analysis of GPT-4's responses on task input/output in a lane coordination with data process (Figure 6-m).

Prompt	**What is the execution of task C conditioned by?**	
Criteria	Case I (response generated by **Ontotext GraphDB 10.8 using OpenAI's GPT-4 model on RDF formatted data**)	Case II (response generated by **GPT-4 via API on BPMN XML formatted data**)
Response	The execution of task C is conditioned by the completion of task B and the availability of data object d1, as indicated by the 'Subsequent' relationship from task B to task C and the 'Data Association' from data object d1 to task C.	The execution of task C is conditioned by the completion of task B, as indicated by the incoming sequence flow from task B to task C in the provided context.
Ground truth	The execution of task C is conditioned by the execution and completion of task B and by data object d1 as input for task C.	
Response relevancy	**0.97**	0.95
Factual correctness	**0.50**	**0.50**
Semantic similarity	**0.96**	0.93
Semantic faithfulness	**0.84**	0.75