LEVERAGING GENERATIVE AI FOR SOUND BUSINESS PROCESS MODEL GENERATION FROM PROCESS NARRATIVES

The 12nd International Conference on Emerging Challenges: Sustainable Strategies in the Data – Driven Economy

Ha Ngoc Long[1], Do Song Huong[1*], Truong Tan Quan[1] and

[1] Faculty of Economics Information System, University of Economics, Hue University, Hue, Vietnam

Faculty of Economics Information System, University of Economics, Hue University, Hue, Vietnam

[*]Corresponding author: dshuong@hueuni.edu.vn

Abstract

This paper addresses a key gap in business process management (BPM) by automating the generation of business process models from unstructured textual descriptions. Traditional BPM methods often rely on manual efforts, making it difficult to capture the complexity of real-world processes. This research proposes a framework that automates the transformation of process narratives into coherent and structurally sound BPM models, significantly reducing manual intervention. The findings show that this approach enhances model accuracy and soundness, streamlining process management and improving integration with existing business systems.

Research purpose:

The purpose of this research is to develop a framework that automates business process model generation from unstructured text, improving model accuracy and efficiency while ensuring logical soundness with minimal manual intervention.

Research motivation:

Current methods for generating business process models from text require extensive manual input and fail to ensure structural correctness. This research addresses the gap by developing an approach that automates the process and ensures model soundness, which is critical for real-world applications.

Research design, approach, and method:

This research follows a design science research methodology, which emphasizes the creation and iterative refinement of a practical framework to automate business process model generation from textual descriptions. The framework is developed through a structured design process that includes problem identification, artifact development, and rigorous evaluation. By drawing on principles of process modeling and leveraging advanced techniques for ensuring structural correctness, the framework is tested across various use cases to assess its effectiveness in reducing manual intervention and improving the accuracy and soundness of the generated models. Multiple cycles of design, evaluation, and refinement are conducted to ensure that the framework meets the requirements of both academic rigor and real-world applicability.

Main findings:

The proposed framework successfully automates business process model generation from textual descriptions while ensuring structural correctness. It reduces the need for manual intervention, improves model accuracy, and addresses challenges of logical consistency, making it applicable in diverse business contexts.

Practical/managerial implications:

Organizations can streamline their BPM lifecycle by automating model generation, reducing manual effort, and ensuring model soundness. This framework enhances process reliability, allowing managers to integrate models with existing systems for more agile and accurate decision-making.

Keywords: Business process management, Process model generation, Artificial intelligence, Large language model, Soundness

1. INTRODUCTION

Business Process Management (BPM) is crucial for organizations aiming to optimize their operations and enhance efficiency. Traditionally, BPM has depended on manually crafted process models, which are often labor-intensive and may not fully capture the complexity and dynamics of real-world processes. To address these limitations, there is a growing need for automated methods to generate process models from unstructured textual descriptions. Textual descriptions, which provide detailed narratives of business processes, offer valuable insights into process execution but are often challenging to convert into structured models (Tomo Licardo, Tanković, & Etinger, 2023). While process mining techniques have emerged, enabling the automatic discovery of process models from event logs. These logs, which capture the sequence of events as they unfold in a process, allow for the generation of models that accurately reflect actual business behavior (Leemans, Fahland, & van der Aalst, 2013).

Textual descriptions, though rich in detail, are typically presented in a narrative format that does not easily translate into structured process models (Friedrich, Mendling, & Puhlmann, 2011b; Kourani, Berti, Schuster, & van der Aalst, 2024). Existing methods for converting these textual narratives into process models often fall short, as they may require extensive manual intervention and are prone to inaccuracies. Traditional BPM techniques and tools, while effective for structured data, struggle with the inherent ambiguity and variability of unstructured text, leading to models that may be incomplete or misrepresentative of the actual processes (Sholiq, Sarno, & Astuti, 2022).

Existing methods for generating BPM models, particularly those not leveraging advanced natural language processing (NLP) or large language models (LLMs), face significant limitations. Traditional approaches often rely heavily on extensive manual input, making them time-consuming and prone to errors. These methods frequently struggle with the variability and complexity of textual data, as conventional NLP techniques may not fully resolve ambiguities or grasp the contextual nuances needed for accurate process modeling. Consequently, this can lead to incomplete or incorrect interpretations of business processes. Additionally, many existing frameworks lack effective mechanisms to ensure the soundness of the generated models, resulting in issues such as deadlocks, inconsistencies, and other anomalies that diminish their practical utility. These methods also generally fail to generalize across various process models, limiting their applicability in diverse business contexts. Furthermore, integrating these models with existing BPM systems is often challenging, as many current approaches do not support seamless integration or provide sufficient insights into process structures. This compromises the reliability and effectiveness of the generated BPM models in real-world applications (Tomo Licardo et al., 2023).

The motivation for this research is grounded in the need to develop a framework for generating BPM models from textual descriptions-an area that remains challenging with current techniques. Traditional BPM methods often require significant manual effort to create accurate models, especially when working with unstructured text. Existing techniques, such as process mining, excel in creating models from event logs but do not translate effectively to text-based inputs. Moreover, many existing methods lack effective mechanisms to ensure the soundness of the generated models, leading to issues like deadlocks, inconsistencies, and other anomalies that undermine their practical utility. Additionally, these methods generally do not generalize well across diverse process models and face challenges in integrating with existing BPM systems, limiting their applicability and effectiveness in real-world scenarios (Tomo Licardo et al., 2023). This gap underscores the necessity for a new approach that can automate the generation of sound BPM models from textual descriptions, reducing the need for extensive manual intervention.

Our primary objective is to develop a framework that generates accurate and sound business process models from textual descriptions. This framework will leverage the capabilities of large language models (LLMs) to process textual descriptions and generate BPM models. Inspired by the structured and iterative nature of process discovery from process mining, our approach will employ a sequence of prompts designed to guide the LLM in extracting, refining, and structuring process information from text. The key advantage of this approach is that it aims to ensure the soundness of the generated models, thereby making the BPM lifecycle more efficient and accessible (Busch, Rochlitzer, Sola, & Leopold, 2023; Leemans et al., 2013; Tomo Licardo et al., 2023).

The proposed framework is developed using a structured approach rooted in the principles of Design Science Research Methodology (DSRM) as defined by Peffers et al. and refined by the design research guidelines of Hevner et al. (Hevner, March, Park, & Ram, 2004; Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). The primary aim is to create a functional artifact (Guideline 1: Design as an Artifact) that addresses the challenge of transforming unstructured textual descriptions into coherent business process models. This process begins by identifying the core difficulty in converting unstructured narratives into structured BPM models, a challenge that existing approaches often fail to address effectively (Guideline 2: Problem Relevance). Through a comprehensive analysis of existing techniques, relevant use cases are defined, ensuring that the proposed framework is grounded in real-world business needs (Guideline 5: Research Rigor).

The design and development of the framework leverage large language models (LLMs) to generate sound BPM models from text. This aligns with the need for more accurate and efficient process modeling methods that minimize manual effort and ensure model correctness (Guideline 3: Design Evaluation). The framework's iterative refinement is guided by

insights from use cases and rigorous textual analysis (Guideline 6: Design as a Search Process). Validation is achieved through a proof-of-concept implementation, testing the system's ability to generate accurate models while addressing errors via both automated and manual feedback loops. The design is continuously improved based on evaluation outcomes, ensuring alignment with BPM principles and practical demands. All steps of the framework's development are thoroughly documented, contributing to both academic research and practical applications in business process modeling and AI (Guideline 4: Research Contributions, Guideline 7: Communication of Research).

The paper is organized as follows: Section 2 reviews the background and related work, covering traditional and AI-based BPM techniques, the role of LLMs in BPM, and the importance of prompt engineering. Section 3 outlines the requirements for the proposed framework, drawing from existing studies. Section 4 explains the proposed framework, showing how it uses prompt engineering inspired by inductive mining to generate BPM models from textual descriptions. Section 5 presents a proof-of-concept implementation, evaluating the framework's effectiveness. Section 6 concludes with a summary of contributions, current limitations, and future research directions.

2. BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Process Model and Process Discovery

This paper adopts the conventions and illustrations of Business Process Model and Notation (BPMN) as established by the Object Management Group (OMG, 2013), as shown in Fig. 1. However, the proposed approach is designed to be flexible and applicable to various graph-based modeling notations. To achieve this versatility, the study uses a graph-based representation of process models. This ensures the concepts and techniques developed are not only aligned with BPMN but also adaptable to other process modeling frameworks, promoting broad applicability and ease of integration in diverse business process management scenarios.

Definition 1. (Process model) A process model P is defined as a connected directed graph. Formally, P is represented as $(N, E, \mathbb{L}, \lambda)$ where:

- N is a finite set of nodes and
- $E \subseteq N \times N$ is a set of edges.
- $\mathbb{L}$ represents the set of labels.
- $\lambda: N \rightarrow \mathbb{L}$ is a labeling function that maps each node to a label in $\mathbb{L}$.

We further define the specific labels as follows:

- $\lambda(n)$ = "Start" and $\lambda(n)$ = "End" are the unique labels denoting the start and end events, respectively.
- $\lambda(n)$ = "XOR-split", $\lambda(n)$ = "AND-split", and $\lambda(n)$ = "OR-split" represent nodes that indicate an exclusive decision, a parallel split, and an inclusive diverging gateway, respectively. These nodes have exactly one incoming edge and a minimum of two outgoing edges.
- $\lambda(n)$ = "XOR-join", $\lambda(n)$ = "AND-join", $\lambda(n)$ = "OR-join" represent nodes that indicate a merge, a parallel join, and an inclusive converging gateway, respectively. These nodes have exactly one outgoing edge and at least two incoming edges.

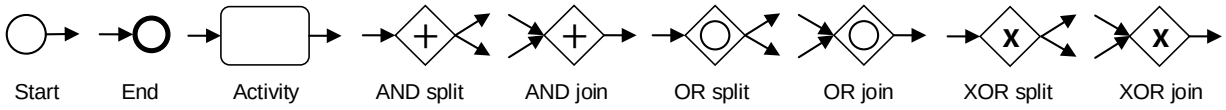

Fig. 1. Types of nodes and their representations

For any node $n \in N$ of a process model $P = (N, E, \mathbb{L}, \lambda)$, The preset of n denote $\bullet n$, includes all the nodes that directly lead to n. It is mathematically defined as $\bullet n = \{m \in N | (m, n) \in E\}$. The postset of n, represented by $n\bullet$, consists of all the nodes that directly follow n. It is mathematically expressed as $n\bullet = \{m \in N | (n, m) \in E\}$.

The semantics of a process model $P = (N, E, \mathbb{L}, \lambda)$ align with those of workflow graphs and similar to the semantics of Petri nets. Typically, their semantics is defined using a *token game*. In this token game a state S: $E \rightarrow \mathbb{N}$ illustrates the token count on each edge. For example, if for an edge e, $S(e) = k$, it implies that e carries k tokens in state S. When executing a node $n \in N$ in the state S, a transition to a new state S' occurs, represented as $S \rightarrow^{n} S'$. A state S' is said to be reachable from state S, notated as $S \rightarrow^{*} S'$, if a finite sequence exists in the form: $S_0 \rightarrow^{n1} S_1 \ldots S_{k-1} \rightarrow^{nk} S_k$, where $k \geq 0$, with $S_0 = S$, $S_k = S'$. The sequence $\langle n_1, n_2, \ldots, n_k \rangle$ denotes an *occurrence sequence* σ of P.

Within the context of process model P, two distinct states stand out in the *initial state*, S_i, and *termination state*, S_o. The initial state S_i is characterized by the presence of a singular token on the outgoing edge from the *Start* node and the absence of tokens elsewhere. On the other hand, the termination state S_o distinguishes itself by housing a singular token

on the incoming edge to the *End* node, with no tokens found elsewhere. An execution sequence of P qualifies as an occurrence sequence σ if and only if *Start* is the opening node and *End* concludes the sequence. Ideally, each execution sequence originates from the initial state and culminates in the termination state.

The notion of soundness, introduced by van der Aalst (Aalst, 1997), was initially defined for workflow nets (a subclass of Petri nets). For workflow graphs (and process models), the definition of soundness simplifies to the absence of the local errors *deadlock* and *lack of synchronization* (Fahland et al., 2011).

Definition 2. (Soundness of a process model P)

- *P* has a *deadlock* iff an AND-join of *P* is active but never enabled in at least one execution.
- P has a *lack of synchronization* iff at least one edge of *P* has more than one token in at least one execution.
- *P* is sound iff it has neither a *deadlock* nor a *lack of synchronization*.

A sound process model is one that is free from logical inconsistencies such as deadlocks or lack of synchronization that could prevent the process from functioning as intended. Soundness is a crucial property in business process management as it ensures that the process can always reach a proper completion, no matter the path taken through the process model. A sound process model guarantees that all tasks can be executed in a way that leads to a valid end state (Aalst, 1997; Fahland et al., 2011). This concept is particularly important in automated process discovery and modeling, where the reliability and correctness of the generated models are paramount (Leemans et al., 2013). Ensuring soundness often involves rigorous validation and verification steps, which can be challenging when dealing with complex or infrequent behaviors. Techniques such as inductive mining are designed to generate sound models by applying systematic partitioning and iterative refinement to the process data (Leemans et al., 2013).

2.1.2 Generative AI: Large Langaguage Models

Generative AI, particularly Large Language Models (LLMs), has revolutionized the field of Natural Language Processing (NLP). LLMs such as OpenAI GPT and Meta Llama are capable of generating human-like text based on large-scale datasets and sophisticated neural network architectures (OpenAI, 2022; Touvron et al., 2023). These models have demonstrated significant potential in automating a wide range of tasks, from content creation to answering complex queries. Recent studies have highlighted the integration of Generative AI, particularly Large Language Models (LLMs), into Business Process Management (BPM), focusing on how these models can automate and optimize process modeling tasks. Researchers have explored the use of LLMs to generate process models from textual descriptions by leveraging techniques like prompt engineering and few-shot learning. For example, in a proposed framework, LLMs are used to convert unstructured text into executable process models through carefully designed prompts and predefined functions, reducing the need for extensive domain expertise and making BPM more accessible to non-experts. The use of these generative models not only accelerates process modeling but also reduces errors through techniques like negative prompting, which instructs the model to avoid common mistakes (Busch et al., 2023; Kourani et al., 2024; Vidgof, Bachhofner, & Mendling, 2023).

Further advancements include the concept of Large Process Models (LPMs), which combine generative AI with neuro-symbolic systems to improve BPM automation. LPMs integrate structured process knowledge with generative AI to provide tailored solutions for different organizational contexts. These models enable the automatic identification of process domains and organizational contexts from unstructured data, offering actionable insights for process design and execution. However, challenges such as the dynamic nature of business processes, the need for continuous retraining, and the scalability of deep learning models still pose significant hurdles. These challenges suggest that while generative AI offers immense potential for BPM, careful consideration of its implementation is necessary to achieve long-term success (Kampik et al., 2024).

2.1.3 BPM life cycle meet LLMs

The integration of Large Language Models (LLMs) into the BPM lifecycle represents a transformative approach that leverages advanced AI to enhance process automation across all phases of BPM. According to Forell et al., LLMs can be employed throughout the BPM lifecycle, including process discovery, design, execution, and continuous improvement (Forell & Schüler, 2024). LLMs play a critical role in converting textual descriptions of business processes into initial process models, thus streamlining the often labor-intensive process discovery phase. Their capability to analyze and interpret unstructured data allows for a more automated and efficient transition from raw process descriptions to structured models (Kampik et al., 2024).

Additionally, the role of LLMs extends to model validation and optimization, where they can suggest improvements based on best practices and industry standards. This ability to provide contextual recommendations enables continuous optimization of processes, enhancing both their efficiency and adaptability. The use of LLMs in BPM is particularly valuable for reducing the reliance on human intervention, as it allows for the automation of routine tasks and quicker responses to evolving business needs. Although this is an emerging area of research, the integration of LLMs into BPM shows considerable promise in improving the scalability, speed, and accuracy of business process management (Grohs,

Abb, Elsayed, & Rehse, 2024; Kampik et al., 2024).

2.2 Related work

Several studies have explored the use of natural language processing (NLP) techniques to generate process models from textual descriptions. Bellan et al. provide a comprehensive overview of techniques for process extraction from text, highlighting various methodologies used in this field (Bellan, Dragoni, & Ghidini, 2020). Gonçalves et al. employed NLP and text mining to derive process models, while Friedrich et al. combined NLP with computational linguistics to generate BPMN models (de A. R. Gonçalves, Santoro, & Baião, 2011; Friedrich, Mendling, & Puhlmann, 2011a). Sholiq et al. further contribute by extracting structured relationship representations from text and converting these into BPMN components, achieving higher accuracy in BPMN generation compared to earlier methods (Sholiq et al., 2022). The BPMN Sketch Miner by Ivanchikj et al. utilizes process mining techniques to create BPMN models from domain-specific language text (Ivanchikj, Serbout, & Pautasso, 2020). Additionally, commercial tools like Process Talks integrate AI to generate process models from textual descriptions (Ivanchikj et al., 2020).

Recent research has also explored the integration of Large Language Models (LLMs) in BPM. Busch et al. and Vidgof et al. investigate the applications and challenges of LLMs in BPM tasks (Busch et al., 2023; Vidgof et al., 2023). Muff and Fill discuss the limitations of GPT-4 in conceptual modeling, while Berti et al. assess LLMs for process mining tasks (Berti, Schuster, & van der Aalst, 2023; Muff & Fill, 2024). Chen and Liao propose using BERT for anomaly detection in process logs (Chen & Liao, 2022). Klievtsova et al. introduce conversational modeling with LLMs for generating process models through dialogue (Klievtsova, Benzin, Kampik, Mangler, & Rinderle-Ma, 2023), and Grohs et al. demonstrate LLMs' capability to translate textual descriptions into process model constraints(Grohs et al., 2024). Fill et al. explore broader implications of LLMs in conceptual modeling (Hans-Georg, Peter, & Julius, 2023).

Prompt engineering has emerged as a critical technique in guiding LLMs, particularly where accuracy and context are paramount. Busch et al. emphasize the importance of crafting effective prompts to ensure LLMs generate relevant and accurate process models (Busch et al., 2023). This technique often involves iterative refinement and chaining prompts to improve coherence and soundness, aligning closely with inductive mining approaches. Kourani et al. present a framework that leverages Large Language Models (LLMs) for automating process modeling from textual descriptions, incorporating advanced techniques in prompt engineering and error handling (Kourani et al., 2024). However, a notable limitation of their approach is its reliance on the Partially Ordered Workflow Language (POWL) for intermediate process representation. While POWL provides robust quality guarantees, it may constrain the flexibility and direct applicability of the generated models. POWL's restriction is that it does not support inclusive behavior, such as OR gateways found in BPMN. This limitation prevents the framework from directly capturing and representing complex decision-making scenarios and alternative paths, which are essential for a comprehensive process model. Addressing this constraint will be crucial for future enhancements to enable the direct generation of detailed BPMN models that accurately reflect diverse process dynamics.

3. REQUIREMENT FOR THE FRAMEWORK

The proposed framework aims to automate the generation of business process models from textual descriptions. This framework is designed to be flexible, accommodating both traditional and advanced approaches. The requirements for this framework are derived from existing research, best practices in process model generation, and the specific objectives of creating sound and accurate process models from text. These requirements are categorized into functional and non-functional categories, ensuring a comprehensive approach to model generation.

3.1 Functional Requirements

3.1.1 Entity and Relationship Extraction (FRQ 1)

The framework must accurately interpret unstructured textual descriptions to extract key entities, such as activities (tasks), events, and participants, and identify the relationships between these entities. It should capture how activities and events are connected, such as which tasks involve specific participants or trigger certain events. For example, in the sentence, "The manager reviews the report, and the team implements the changes," the framework should extract "review" and "implement" as activities, and identify the relationships, such as "the manager" being responsible for "reviewing" and "the team" for "implementing". This ensures a clear understanding of the entities and their relationships, forming the basis for accurate process modeling (Busch et al., 2023; Tomo Licardo et al., 2023).

3.1.2 Process Structure and Control Flow Extraction (FRQ 2)

The framework must accurately identify and extract the overall process structure and control flow from textual descriptions. This includes:

- Sequential Flow: Ensuring the correct order of tasks to reflect the process flow accurately.
- Parallel and Conditional Branching: Detecting and modeling parallel tasks and decision points for accurate representation.

- Loop Detection: Identifying and modeling loops where tasks repeat until conditions are met.
- Hierarchy and Nesting: Recognizing hierarchical and nested processes to accurately represent sub-tasks within larger activities.
- Gateway Handling: Identifying and representing gateways (e.g., XOR, AND, OR) to control process flow based on decisions.

This comprehensive extraction ensures that all aspects of the process are correctly modeled, reflecting the true dynamics of the business process (Busch et al., 2023; Tomo Licardo et al., 2023).

3.1.3 Soundness (FRQ 3)

Ensuring the soundness of generated process models is a non-negotiable requirement. The framework must include an automatic evaluation mechanism to verify the logical correctness of the models generated by the LLM or NLP system. This includes checking for deadlocks, livelocks, and other anomalies. If any issues are detected, the framework should trigger prompt adjustments and reprocessing until a sound model is achieved. This process aligns with soundness verification principles from inductive mining (Choi, Kongsuwan, Joo, & Zhao, 2015; Leemans et al., 2013; Prinz, Prinz, & De, 2017; van der Aalst, Hirnschall, & Verbeek, 2002).

3.1.4 Iterative Refinement and CRUD Operations (FRQ 4)

The framework should support iterative refinement of business process models through the effective implementation of CRUD (Create, Read, Update, Delete) operations. If the model generated by the LLM or NLP system is not accurate or sound, the CRUD capabilities will facilitate the refinement process by allowing the framework to create a new model, update the existing model based on refined prompts or parsing logic, and continuously reprocess the textual description until a satisfactory model is achieved. This iterative approach ensures continuous improvement and alignment with the intended business process (Busch et al., 2023; Kourani et al., 2024; Leemans et al., 2013).

In this framework, CRUD operations are not just standalone functionalities but integral tools that enhance iterative refinement. Users can create new models directly from textual descriptions, retrieve and review existing models to identify discrepancies, update models with revised inputs to correct inaccuracies, and delete models that are no longer relevant. These operations are seamlessly integrated, enabling users to efficiently manage and refine business process models as the underlying textual information evolves. The design prioritizes ease of use, allowing users to transition effortlessly between creating, reading, updating, and deleting process models, thereby supporting the iterative refinement process in response to changing business requirements (Busch et al., 2023; Tomo Licardo et al., 2023).

3.1.5 Model Generation and Visualization (FRQ 5)

The framework must be capable of generating a generic process model from the extracted information, ensuring that it is not strictly tied to any specific formal language, such as Petri nets, EPCs, workflow graphs, or BPMN. This approach allows the generated model to be easily transformed into any desired formal language later. The visualization should still incorporate essential elements like tasks, events, gateways, and flows to ensure that the process is clearly represented. The framework should produce a generic process model that accurately reflects the sequence of tasks, decision points, and parallel activities described in the text, while remaining adaptable for transformation into various formal representations as needed (Busch et al., 2023; Tomo Licardo et al., 2023).

3.1.6 Integration with BPM Tools (FRQ 6)

For practical deployment, the framework must ensure that the generated business process models are compatible with existing BPM tools. This includes the capability to export models in standard formats, such as BPMN, and provide APIs or other interfaces to facilitate seamless integration with BPM software. By supporting these standard formats and integration methods, the framework enables easy adoption within existing business processes and toolchains, ensuring that the generated models can be effectively utilized in real-world applications and workflows (Busch et al., 2023).

3.2 Non-Functional Requirements

3.2.1 Scalability and Performance (NFRQ 1)

The framework must maintain high performance while processing large datasets and complex descriptions. It should scale effectively to handle varying levels of complexity without compromising accuracy or speed, which is crucial in dynamic business environments where rapid model generation is necessary (Kourani et al., 2024; Ouyang, Adams, ter Hofstede, & Yu, 2018)

3.2.2 Adaptability and Customization (NFRQ 2)

The framework should be adaptable to different business domains and allow customization of the process model generation according to specific domain requirements. This includes flexibility to tweak extraction rules, model generation parameters, and visualization settings based on industry-specific needs. Adaptability ensures the framework's

broad applicability across different industries and use cases (Busch et al., 2023; Kourani et al., 2024).

3.2.3 User Validation and Feedback (NFRQ 3)

The framework should provide interfaces for user validation and feedback, allowing users to review and correct the generated process model. This interaction ensures that the model aligns with the user's understanding and expectations. After generating a BPMN model, the user should be able to inspect, adjust, and provide feedback on the model, which the framework can learn from. User validation is essential for refining the model, ensuring its practical utility, and fostering user trust in the automated process model generation system (Forell & Schüler, 2024; Kourani et al., 2024).

3.3 Design Considerations

As we develop the proposed framework, several critical design decisions and challenges need to be considered to ensure the system's robustness, flexibility, and practical applicability across various business domains.

3.3.1 Design Decisions (DD):

Key decisions include determining the balance between automation and user control, particularly in handling ambiguities in textual descriptions. The integration with existing BPM tools is crucial for ensuring compatibility and ease of adoption. Additionally, the framework must decide between leveraging external API services for LLMs versus deploying a local LLM instance, each with distinct trade-offs in cost, computational resources, and data privacy. Another important decision revolves around the iterative refinement process, ensuring continuous improvement of the generated models (Busch et al., 2023; Grohs et al., 2024; Kourani et al., 2024; Leemans et al., 2013).

3.3.2 Design Challenges (DC):

The framework will face challenges such as managing ambiguity and inconsistency in textual descriptions, ensuring scalability when processing large volumes of text, and integrating with diverse BPM tools. Moreover, ensuring the soundness of generated models and gaining user trust are critical challenges that must be addressed to ensure the framework's effectiveness and reliability. Managing costs associated with using external APIs versus local deployment presents an additional layer of complexity, particularly in enterprise environments (Grohs et al., 2024; Kourani et al., 2024).

4. THE FRAMEWORK

The framework for generating accurate and sound business process models from textual descriptions is a multi-stage approach that leverages the capabilities of large language models (LLMs) to automatically generate business process models from textual descriptions. This framework not only facilitates process model generation but also includes verification and feedback mechanisms to ensure the model's accuracy and correctness.

The framework's workflow begins with a natural language input, referred to as the *Input Process Narrative*, which describes a business process. The system first performs *Context Understanding*, parsing the input to identify the process goals and establish its context. From there, *Entities Identification* ensures all unique activities, actors, and tasks are recognized, eliminating any redundancies or duplications. The framework proceeds with *Gateways Identification*, pinpointing decision points and parallel flows that determine alternative paths within the process. *Loops Identification* detects any repeated tasks, ensuring accurate representation of iterative activities. After identifying all these elements, the system organizes them through *Sequence Flows Identification*, creating a coherent process flow that aligns with business objectives.

Once the process flow is established, the framework moves into *Business Process Generation*, where it transforms the identified elements into a formal business process model using Graphviz's DOT language. The generated model then undergoes *Business Process Verification* to ensure structural correctness and adherence to BPM principles. If errors are detected, the framework either triggers an automated feedback loop for corrections or requests manual input from process experts. Finally, after all necessary adjustments are made, the verified business process model is exported for integration into business systems. This workflow highlights the iterative approach of SProLLaM, blending automation and feedback to create accurate and reliable process models. This workflow, depicted in Fig. 2, highlights the iterative approach of SProLLaM, blending automation and feedback to create accurate and reliable process models

4.1 Components of the framework

The framework is structured into two main stages: Automatic Process Model Generation and Automatic Structural Verification of Business Process. The framework incorporates a series of systematic prompts, services, and feedback loops that work together to ensure accuracy, soundness, and coherence throughout the business process model generation lifecycle. It employs prompt chaining to leverage multiple LLM models, offering flexibility and cost efficiency (*NFRQ 3*), while maintaining scalability and performance (*NFRQ 1*).

4.1.1 Stage 1 - Automatic Process Model Generation

This stage focuses on generating a business process model from an input process narrative, which typically consists of textual descriptions of business activities. The framework uses NLP techniques to parse and extract key elements from the text (*FRQ 1*), handling linguistic variations and ambiguities to accurately interpret complex or poorly structured inputs. This stage aligns with the framework's functional requirements for input processing, entity extraction, and process structure extraction (*FRQ 1, FRQ 2*).

- *Context Understanding (FRQ 1):* The process begins with interpreting the text to understand the overall goals and context of the business process. This component uses NLP techniques for syntax parsing and semantic analysis to identify key objectives, which guide the subsequent steps in model generation. For example, it ensures that tasks like "*approval*" and "*forwarding*" are correctly identified from the phrase "Once the document is approved, it is forwarded to the manager."
- *Entities Identification (FRQ 1):* Once the context is understood, the framework identifies distinct activities, actors, and tasks within the process. This step ensures that all entities are unique and that no duplications or redundancies exist. It also establishes relationships between tasks, recognizing dependencies, sequences, and parallel flows, ensuring that the model reflects the true dynamics of the business process.
- *Gateways Identification (FRQ 2): This* component recognizes decision points, parallel flows, and branches within the process. It identifies conditions that lead to different paths in the process, such as decision points like "if" conditions or branching logic. For example, in a phrase like "*If the budget is approved, proceed with procurement; otherwise, revise the proposal,*" this component ensures that these alternative paths are identified and correctly modeled.
- *Loops Identification (FRQ 2):* The framework detects any loops or repeated tasks in the process. It identifies activities that repeat until certain conditions are met, such as a phrase like "*repeat the inspection until the issue is resolved*." Accurately modeling these loops is critical for representing iterative processes within the business flow.
- *Sequence Flows Identification (FRQ 2):* After all tasks, entities, gateways, and loops are identified, this component organizes them into a coherent sequence flow. It ensures that the business process flows logically from one step to the next, respecting all identified dependencies and conditions. This stage ensures that the generated process model accurately reflects the described process's flow, hierarchy, and control logic, aligning with business objectives.

4.1.2 Stage 2 - Automatic Structural Verification of Business Process

Once the initial process model is generated, Stage 2 focuses on ensuring the model's structural soundness and alignment with business rules (*FRQ 3*). This stage involves verifying the logical correctness of the generated model using external tools, ensuring it is free from issues such as deadlocks and livelocks, and that it adheres to BPM principles. This stage addresses the functional requirement for model generation and soundness verification, ensuring that the business process model adheres to established principles of BPM and process mining (*FRQ 3, FRQ 5*).

- *Business Process Generation (FRQ 5):* The framework transforms the identified sequence flows and activities into a formal business process model. This model is typically generated in a generic format, such as Graphviz's DOT language, allowing for adaptability and subsequent transformation into other formal languages like BPMN or Petri nets (*FRQ 5*). This flexibility ensures that the model can be integrated with various BPM tools (*FRQ 6*).
- *Business Process Verification (FRQ 3):* After generating the model, this component verifies its structural correctness and logical soundness. External verification tools are employed to check for errors such as deadlocks, improper task sequences, and other anomalies that could disrupt the execution of the process model. The framework ensures that the process flows correctly, with all conditions, branches, and loops functioning as intended.
- *Evaluation (FRQ 3):* If errors or inconsistencies are detected during the verification process, the framework evaluates these discrepancies and suggests corrections. The evaluation process identifies areas that require adjustments, ensuring that the model is logically consistent and aligned with the business rules and objectives.

4.1.3 Error Handling and Feedback Loops

To ensure that the generated business process model is accurate and reliable, the framework incorporates both automated and manual feedback loops, enabling iterative refinement (*FRQ 4*). These loops allow for continuous improvements until a sound and correct model is achieved. This stage fulfills both the functional and non-functional requirements for iterative refinement, model export, and user validation (*FRQ 4, FRQ 6, NFRQ 3*)

- *Auto Feedback Loop (FRQ4):* This loop is triggered automatically when the verification process detects errors. Predefined prompts suggest corrections for common syntax and semantic issues within the model, ensuring that minor issues are resolved quickly without requiring human intervention. The framework leverages prompt chaining to reprocess the text and adjust the model iteratively, enhancing cost efficiency (*NFRQ 2*).

- *Human Feedback Loop (FRQ 4, NFRQ 3):* When automated corrections are insufficient or ambiguous, this component enables manual feedback from process experts. Users can inspect, review, and make adjustments to the model, ensuring that it aligns with their understanding and expectations. This interaction fosters user trust in the system and ensures the model's practical utility.
- *Export Business Process (FRQ 6):* Once the model has been verified and refined, the final business process model is exported in a standard format, such as BPMN, for integration into business systems. This output is compatible with existing BPM tools, enabling seamless deployment in real-world applications (*FRQ 6*).

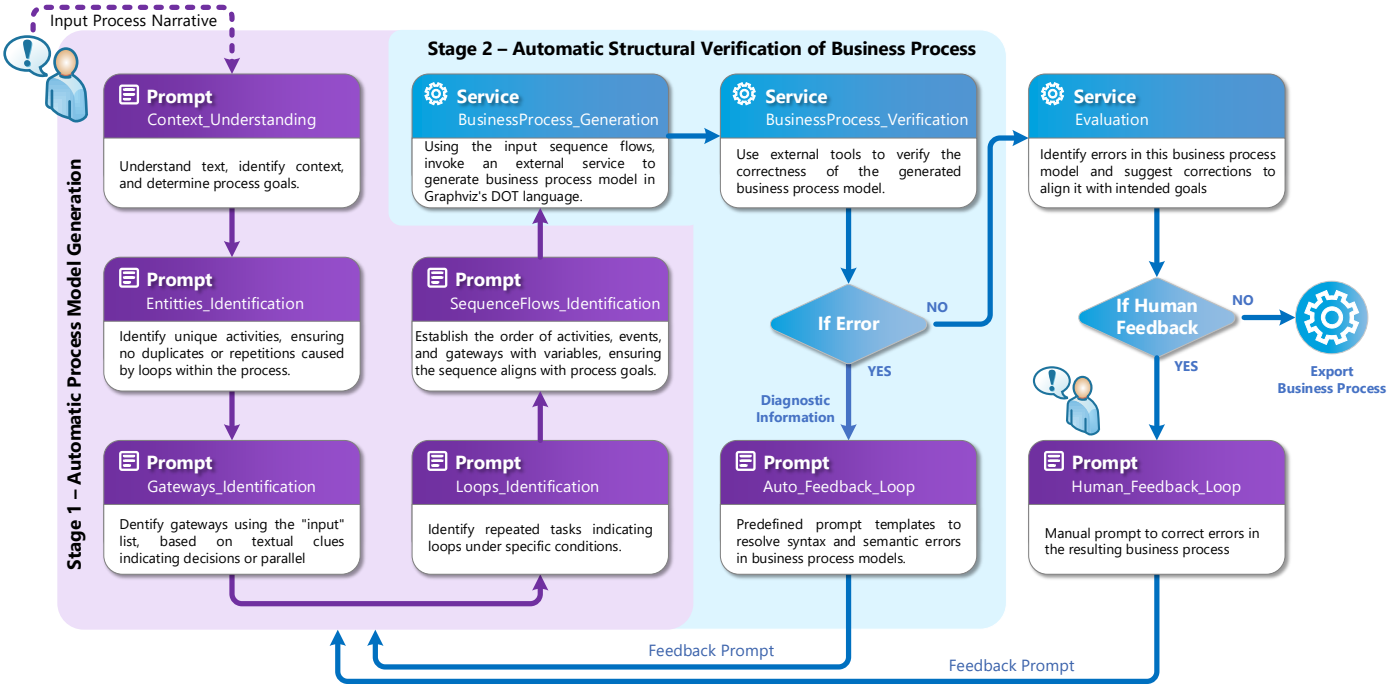

Fig. 2. The framework for sound process model generation using large language models

4.2 Prompt Engineering for Automated Process Model Generation

This section delves into the strategies and methodologies integral to the framework for generating accurate business process models from natural language descriptions. Key techniques include prompt engineering methods such as prompt chaining and chain-of-thought reasoning. These techniques guide large language models (LLMs) to accurately interpret and model complex business processes. Prompt engineering focuses on crafting prompts that elicit precise responses while leveraging the strengths and managing the limitations of LLMs. By breaking down complex tasks into smaller, sequential prompts, prompt chaining creates a workflow where each step builds on the last. This approach allows for nuanced guidance, ensuring the model handles intricate queries effectively. These strategies enable efficient model generation while adhering to principles of flexibility, cost-effectiveness, and iterative refinement (OpenAI, 2022; Touvron et al., 2023)

4.2.1 Prompt Engineering

The framework incorporates a range of advanced prompt engineering techniques to maximize the performance of LLMs in business process model generation. These techniques include prompt chaining, chain-of-thought reasoning, knowledge injection, and few-shots learning to improve accuracy and efficiency:

- *Decomposition of Tasks (Chain-of-Thought Reasoning):* The framework uses chain-of-thought reasoning to break down the task of process model generation into smaller, sequential steps. Each prompt focuses on a specific subtask, such as understanding the context, identifying actions, or recognizing gateways. By guiding the LLM step by step through these subtasks, the framework ensures that the generated process models are coherent and logically consistent. This approach reduces the cognitive load on the LLM and helps it generate accurate outputs, even for complex business processes.
- *Prompt Chaining for Structured Model Generation:* Prompt chaining is central to the framework's strategy. Each prompt's output serves as the input for the next prompt, creating a flow of information that progressively refines the process model. For example, after the *Context Understanding* prompt establishes the process scope and objectives, its output is used by the *Entities Identification* prompt to extract specific tasks and events. This

chaining allows the model to build incrementally, ensuring that each aspect of the process is handled in a logical and structured manner.

- *Knowledge Injection for Domain-Specific Understanding:* The framework leverages knowledge injection to provide the LLM with domain-specific insights that may not be part of its pre-existing knowledge base. For example, the framework can inject detailed knowledge about BPMN 2.0.2 standards, business rules, or specific industry practices to ensure that the generated models adhere to the required standards. By enhancing the LLM's understanding of these specialized concepts, knowledge injection reduces the likelihood of errors and ensures that the output models are contextually appropriate and compliant with industry norms.
- *Few-Shots Learning for Enhanced Performance***:** Few-shots learning is employed to further improve the LLM's performance in generating process models. The framework provides the LLM with a few examples of process descriptions along with their corresponding expected models. These examples help the LLM better understand the structure and logic required in process modeling, enabling it to generate more accurate outputs. By giving the LLM relevant examples, the framework enhances its ability to handle similar tasks and scenarios, ensuring that the output models align with best practices.
- *Role-Based Prompting for Expertise:* The framework uses role-based prompting to instruct the LLM to assume specific roles during model generation. For example, the LLM is prompted to act as a process modeling expert, ensuring that it applies industry knowledge and best practices when generating the process model. This approach ensures that the LLM produces high-quality, contextually appropriate models by leveraging its role-specific expertise.
- *Negative Prompting for Error Prevention:* Negative prompting is used to instruct the LLM on what to avoid during model generation. This includes avoiding common errors, such as incorrect loop structures or improper gateway configurations. By preemptively guiding the LLM away from these pitfalls, the framework ensures a higher level of accuracy in the output, reducing the need for manual corrections later in the process.

These prompting strategies, combined with chain-of-thought reasoning, prompt chaining, knowledge injection, and few-shots learning, ensure that the framework generates accurate, sound, and contextually relevant business process models.

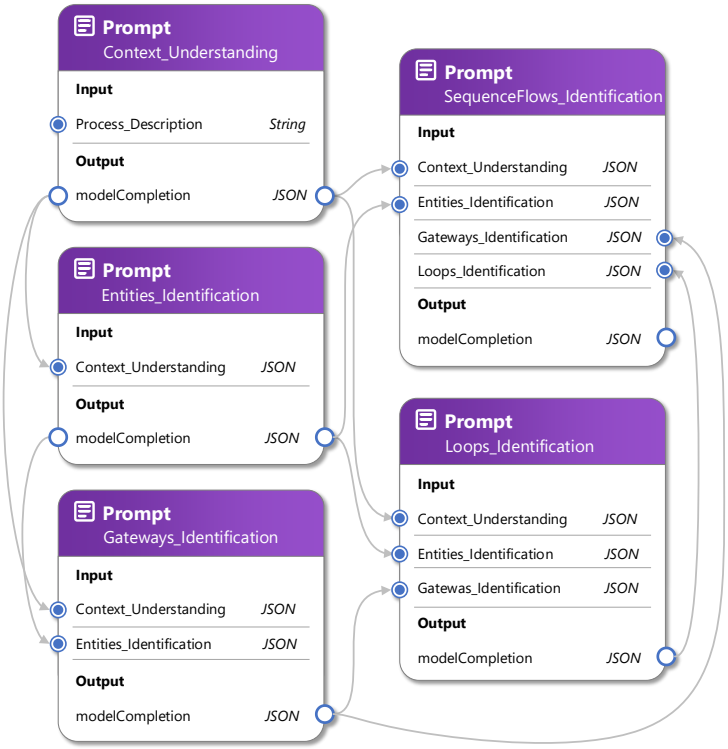

Fig. 3. Prompt Flows for automated process model generation

4.2.2 Process Model Generation

Once the LLM generates the core elements of the business process model, the framework transitions into the process model generation phase. This phase focuses on producing a **generic process model** that supports key business process elements, ensuring that the model is adaptable for various practical applications.

- *Generating the Generic Process Model:* The framework generates a generic process model that incorporates essential elements such as tasks, events, gateways, loops, and sequence flows. The model is output in a flexible

format, specifically as a Graphviz DOT file. This format allows the model to be visualized and further refined as needed, making it a practical solution for real-world business scenarios.

- *Visualization of the Process Model:* After the model is generated in Graphviz DOT format, it is visualized using graph-based visualization tools. This visualization provides a clear representation of the process flow, allowing users to easily analyze and validate the model. The visualization includes all core elements of the process, ensuring that the model is easy to interpret and aligned with the business goals.
- *Export and Integration:* The framework supports exporting the generated process model in standard formats such as BPMN, ensuring compatibility with existing BPM tools. This export capability allows businesses to integrate the generated models into their existing workflows, making the framework a versatile and practical tool for business process management.
- *Refinement Loop with User Feedback:* The framework incorporates an interactive refinement loop that allows users to provide feedback on the generated model. This feedback is processed through additional prompts, which refine the model based on user input. The feedback loop ensures that the model continues to evolve and adapt to specific business requirements, providing a continuous improvement process that aligns with user expectations.

Through the process model generation phase, the framework ensures that the generated models are practical, adaptable, and ready for deployment in real-world business scenarios. By generating a generic process model, visualizing it, and integrating it into business systems, the framework offers a comprehensive solution for business process modeling.

4.3 Feedback loops

Just as humans can make errors when modeling complex business processes, large language models (LLMs) may also produce inaccuracies in process models, despite their advanced capabilities. To address these issues, the framework integrates feedback loops to ensure that generated business process models are accurate, logically sound, and aligned with business objectives. These feedback loops operate in both automated and manual capacities, addressing potential errors and facilitating iterative refinement. By leveraging these loops, the framework ensures that the models evolve toward higher precision and quality.

4.3.1 Adjustable and Non-Adjustable Errors

Errors in business process models can be categorized into two types: *adjustable errors* and *non-adjustable errors*. The framework manages these errors through iterative LLM adjustments or escalates them for human intervention, depending on the complexity and nature of the issue.

- *Adjustable Errors:* These are errors that the system can resolve through iterative engagement with the LLM. They typically involve minor syntax inconsistencies, synchronization misalignments, or gateway misconfigurations. For example, if a misconfigured gateway causes a potential deadlock, the system generates prompts to modify the gateway type and ensure proper flow. Most adjustable errors are resolved efficiently within a few iterations, minimizing the need for manual input.
- *Non-Adjustable Errors:* These are more complex issues that cannot be resolved through automated LLM adjustments alone. They include significant structural flaws, such as deadlocks, irreducible loops, or major synchronization problems that persist after several iterations of corrective prompts. When these issues occur, the system escalates them for human intervention. The system halts further automated corrections and flags the error for human review, providing detailed diagnostic information to guide manual resolution.

This categorization allows the framework to efficiently address a wide range of errors autonomously while ensuring that more complex and critical problems are escalated to experts when necessary

4.3.2 Refinement Based on Verification Results

The framework integrates a robust automatic error-handling mechanism that detects and resolves adjustable errors during the verification stage. The framework relies on verification rules (Choi et al., 2015), such as ensuring structural consistency, correcting loop configurations, and resolving synchronization issues, to guide the LLM in refining the model. These rules target critical elements like loop gateways (e.g., headers, exits, and backward splits), helping to detect and correct deadlocks, synchronization issues, and other structural conflicts (Choi et al., 2015).

While the framework is based on the verification rules from (Choi et al., 2015), it is designed to be adaptable, allowing new diagnostic information and verification rules to be integrated with minimal modification. By leveraging diagnostic information, the framework identifies and addresses control-flow errors that could obstruct the correct functioning of business process models. Engaging the LLM in an iterative feedback loop, the system dynamically corrects these errors, ensuring that the models conform to BPMN standards and business logic. The paper will detail various types of diagnostic information, such as syntax errors, deadlocks, and lack of synchronization, along with methods for their automatic resolution, ensuring the generated models are ready for practical use in simulation, code generation, and execution.

Table 1. Types of Diagnostic Information

Syntax Errors	Deadlocks	Lack of Synchronization
Syntax errors in process models include issues such as activities or events having multiple incoming or outgoing edges, gateways not being explicitly defined as split or join, and gateways with only a single incoming and outgoing edge. Additionally, models lacking start or end events are considered syntax errors.	A deadlock occurs when a token becomes "*stuck*" on an edge of an AND-join, preventing further progress in the process.	A lack of synchronization occurs when multiple tokens appear on an edge that should have only one token. This typically arises from improperly merged parallel paths, such as merging parallel paths with an XOR-join, leading to potential inconsistencies in execution.

Below are generalized prompt templates with placeholders to illustrate how the framework addresses these errors, aligned with specific verification rules:

Ensuring structural consistency in the acyclic process (VR4): During the verification process, the system ensures that the final acyclic model is free from structural conflicts that could lead to deadlocks or synchronization issues. These conflicts often arise when gateways are misconfigured, causing tokens to become stuck or accumulate improperly, which can disrupt the process flow.

- Prompt Template:
  "Structural conflicts have been detected at the [*current_gateway_type*] gateway in the acyclic model, which could result in [*issue_type*] (e.g., deadlock or synchronization issue). Please review the gateway configuration and reconfigure it as an [*alternative_gateway_type*] gateway to ensure smooth process flow and prevent token blockage or accumulation."
- Placeholders:
  [*current_gateway_type*] - The type of gateway currently causing the issue (e.g., AND).
  [*issue_type*] - The type of issue detected (e.g., deadlock, synchronization issue).
  [*alternative_gateway_type*] - The suggested gateway type to replace the current configuration (e.g., XOR).

Correcting loop gateways (VR1): When issues arise at loop headers, exits, or backward splits—such as deadlocks or synchronization problems caused by improper gateway configurations—the system generates a prompt to adjust the control flow:

- Prompt Template:
  "The current configuration at the loop [*header/exit/split*] in [*loop_position*] is causing issues with process flow. Please change the [*current_gateway_type*] gateway to an [*alternative_gateway_type*] gateway to ensure proper synchronization and prevent deadlocks."
- Placeholders:
  [*loop_position*] - The specific position within the loop where the issue occurs.
  [*current_gateway_type*] - The type of gateway currently causing the issue (e.g., AND).
  [*alternative_gateway_type*] - The suggested gateway type to replace the current configuration (e.g., XOR).

Resolving conflicts in extended forward and backward flows (VR2): If structural conflicts are detected in the extended forward flow (eFwd) or extended backward flow (eBwd) of a natural loop, the system engages the LLM with a prompt to verify and adjust the configuration of nodes and edges:

- Prompt Template:
  "Structural conflicts have been identified in the extended flow between [*start_node*] and [*end_node*] in the loop. Please review and adjust the configuration of nodes and edges to ensure no concurrency issues or conflicts exist."
- Placeholders:
  [*start_node*] - The starting node of the conflicted flow.
  [*end_node*] - The ending node of the conflicted flow.

Fixing multiple exits leading to the same node (vr3): When multiple loop exits converge on the same destination node outside the loop, a prompt is generated to reconfigure the gateway at the destination node to prevent unintended multiple instantiations:

- Prompt Template:
  "Multiple exits from the loop are converging on the node at [*destination_node*], which could result in unintended multiple instantiations of the process. Please adjust the configuration by setting the [destination_node] gateway to an [*alternative_gateway_type*] join to prevent duplication of paths."
- Placeholders:
  [*destination_node*] - The node where multiple exits are converging.
  [*alternative_gateway_type*] - The suggested gateway type (e.g., XOR) to manage the converging paths.

Handling irreducible loops with multiple entries and exits (vr5): For more complex irreducible loops that have multiple entry and exit points, the system prompts the LLM to adjust the gateways to prevent deadlocks and other issues:

- Prompt Template:
  "An irreducible loop with multiple entry/exit points has been detected at [*loop_position*]. Please modify the gateway configuration by replacing the [current_gateway_type] gateways with [*alternative_gateway_type*] gateways at the loop entries/exits to ensure proper flow and prevent deadlocks."
- Placeholders:
  [*loop_position*] - The position of the irreducible loop.
  [*current_gateway_type*] - The type of gateway currently causing issues at the loop entries/exits.
  [*alternative_gateway_type*] - The suggested gateway type to resolve the issue (e.g., XOR).

These prompt templates provide the LLM with the necessary guidance to dynamically correct the detected errors in the business process model. By iteratively refining the model based on the provided diagnostic information, the framework ensures that the generated models are robust, accurate, and free of structural conflicts. The use of placeholders allows these prompts to be adaptable across a wide range of scenarios, ensuring flexibility in addressing various verification challenges.

4.3.3 Refinement Based on Human Feedbacks

For non-adjustable errors that the automated system cannot resolve, human feedback becomes crucial in refining the business process models. These non-adjustable errors are flagged for manual review by process experts, who are equipped with decision support information to aid in correcting the model. Human intervention ensures that even the most complex or ambiguous issues are addressed, preserving the quality and accuracy of the final process model.

- *Interactive Refinement:* Once the automatic system identifies non-adjustable errors, the process is handed over to human experts. These experts are provided with detailed diagnostic information, including the error type, location, and previous automated attempts at resolution. This information allows them to make informed decisions when refining the model. The framework supports interactive refinement, where human feedback is incorporated back into the system for iterative improvements.
- *Decision Support Information:* The framework offers decision support information to help human experts understand the implications of their changes. This includes visual representations of the problematic areas, suggestions based on BPMN standards, and historical context from previous iterations. By providing this information, the system ensures that experts can make precise adjustments while considering the broader impact on the process model.
- *Feedback Loop Integration:* After human corrections are made, the framework integrates this feedback into the model generation pipeline, allowing the system to learn from these adjustments. The updated model undergoes another round of verification to ensure that the manual corrections have resolved the issues without introducing new problems. This loop of human feedback and automated verification ensures continuous improvement of the model, aligning it with business goals and technical requirements.

Through a combination of automated refinement and human intervention, the framework ensures that business process models are both accurate and practical for real-world applications. The balance between LLM-driven automation and expert-driven manual refinement guarantees that even complex, non-adjustable errors are resolved effectively

4.4 Design Decisions (DD)

The development of the framework necessitates several critical design decisions to balance the framework's flexibility, robustness, and adaptability across different business domains. These decisions directly impact the framework's usability, performance, and scalability. Key considerations include the balance between automation and user control, integration with existing BPM tools, handling ambiguities in textual descriptions, the iterative refinement process, and managing computational resources.

4.4.1 DD1: Level of Automation vs. User Control

One of the foundational design decisions in the framework is determining the balance between automation and user control. The framework aims to automate the process model generation from textual descriptions as much as possible, minimizing the need for manual intervention. However, complete automation may not be feasible or desirable in all cases, especially when dealing with complex or ambiguous texts.

Allowing user control at critical junctures in the process can help ensure that the generated models accurately reflect the business processes as understood by human experts. For example, when the LLM encounters ambiguities in the text, the framework could either attempt an automated resolution based on predefined rules or prompt the user for clarification. Configurable settings within the framework should allow users to adjust the level of automation and intervention, enabling them to step in when necessary. This balance between automation and user control directly influences the framework's usability, as more automation reduces user workload, but excessive automation might lead to inaccuracies if the model

does not align perfectly with user expectations (Busch et al., 2023)

4.4.2 DD2: Integration with Existing BPM Tools

Integrating the generated process models with existing BPM tools is another crucial design decision. The framework must ensure compatibility with widely used BPM platforms, which often support specific formats, such as BPMN, XPDL, or Petri nets. The choice of output formats and data interchange mechanisms will influence how easily organizations can adopt the framework and integrate it into their existing workflows.

Deciding on the level of compatibility with existing BPM tools also impacts how seamless the integration will be. For example, providing APIs and export options in standard BPM formats can facilitate adoption without requiring significant customization or re-engineering of current systems. This decision is vital to the framework's practical deployment, as successful integration with BPM tools is essential for real-world applications (Busch et al., 2023)

4.4.3 DD3: Handling of Ambiguity in Textual Descriptions

Textual descriptions of business processes often contain ambiguities and inconsistencies. The framework must decide on how to handle these ambiguities to ensure accurate and reliable process model generation. Options include leveraging predefined rules, allowing user feedback loops, or incorporating advanced natural language processing (NLP) techniques to disambiguate and clarify the text autonomously.

This decision affects the framework's capability to deal with real-world text, which is often vague or incomplete. Advanced NLP techniques, such as context-aware language models, can help resolve ambiguities automatically, but there may still be cases where human input is necessary. The framework needs to be designed with flexibility in mind, enabling it to escalate complex cases for manual review when automated methods are insufficient (Tomo Licardo et al., 2023; Busch et al., 2023)

4.4.4 DD4: Iterative Refinement Process

The iterative refinement process is a critical design decision that determines how the framework continuously improves the generated process models. This involves deciding on the number of iterations allowed, the criteria for determining when a model is sufficiently refined, and how user feedback is integrated into subsequent iterations.

The framework should be designed to support iterative refinement, allowing for continuous model adjustments based on feedback from automated verifications or user input. This iterative approach ensures that the final process models are sound and align with business objectives. The refinement process must be flexible enough to accommodate varying levels of complexity in the input text and provide mechanisms for continuous improvement (Leemans, Fahland, & van der Aalst, 2013).

4.4.5 DD5: Cost and Computational Resources

A critical design decision in this framework involves determining how to allocate computational resources effectively between using external API services for LLMs (such as GPT-based models provided by OpenAI (OpenAI, 2022)) or deploying a local LLM instance (Touvron et al., 2023). External APIs offer the advantage of simplicity and access to powerful, continuously updated models, but they come with recurring costs that can escalate significantly with increased text processing volumes. Conversely, deploying a local LLM requires substantial upfront investment in computational resources, such as powerful GPUs or specialized hardware, but can lower long-term operational costs and provide greater control over data privacy.

The design decision must also account for the modular nature of the framework, where different approaches can be applied to different steps based on their importance and resource needs. For the most critical steps that demand high accuracy and sophisticated language processing, more resources, such as LLMs, will be allocated, potentially through API services. In contrast, less critical steps might employ local LLMs or traditional NLP techniques, which are less resource-intensive but still sufficient for those tasks. This strategic allocation of resources ensures that each step in the process is handled cost-effectively, balancing performance, scalability, data security, and budget constraints (Tomo Licardo et al., 2023).

4.5 Design Challenges (DC)

The development of the framework involves overcoming key design challenges related to ensuring its effectiveness, scalability, and reliability across diverse business domains. Addressing these challenges is essential for successful deployment and adoption in real-world scenarios.

4.5.1 DC1: Ambiguity and Inconsistency in Textual Descriptions

One of the primary challenges in generating business process models from textual descriptions is handling the inherent ambiguity and inconsistency often present in natural language. Business process narratives are frequently vague, incomplete, or open to multiple interpretations, making it difficult to generate accurate and reliable process models automatically. To address this challenge, the framework must incorporate advanced methods that can interpret and clarify

ambiguous text. This may include using context-aware analysis, semantic understanding, and domain-specific knowledge to resolve ambiguities. However, even with sophisticated techniques, some cases will still require human intervention. Balancing automation with opportunities for user feedback and manual correction is essential to overcoming this challenge and ensuring the generated models align with real-world business processes (Tomo Licardo et al., 2023).

4.5.2 DC2: Scalability in Large-Scale Implementations

Scalability is another significant challenge for the framework, particularly when processing large volumes of text or handling highly complex business processes. As the size and complexity of the input data increase, the framework must maintain high performance without sacrificing accuracy or speed. To address this, advanced methods must be implemented to optimize the framework's architecture, enabling it to efficiently process larger workloads. This may involve techniques such as distributed computing, parallel processing, and horizontal scaling, which allow the framework to expand its capacity by adding computational resources as needed (Busch et al., 2023).

4.5.3 DC3: Integration with Diverse BPM Tools

Integrating generated process models with various BPM tools presents a complex challenge due to the diverse standards and practices across industries. Different BPM platforms often use distinct data formats, modeling languages, and workflows, making seamless integration difficult without extensive customization. To address this, the framework must be designed to be flexible and adaptable, supporting multiple output formats and providing integration mechanisms such as APIs. This flexibility is essential to ensure compatibility with a wide range of BPM tools, allowing organizations to adopt the framework regardless of their existing BPM infrastructures and workflows. (Busch et al., 2023).

4.5.4 DC4: Ensuring Model Soundness

Ensuring the soundness of generated process models is a critical challenge, especially when dealing with complex processes or ambiguous textual descriptions. A sound process model must be free from logical errors such as deadlocks, improper task sequencing, or synchronization issues that could disrupt the intended process flow. To address this, the framework must incorporate robust verification mechanisms capable of automatically detecting and correcting logical errors. These mechanisms need to handle a wide range of potential issues, from basic syntax mistakes to intricate structural conflicts. Implementing such verification processes in a manner that is both thorough and efficient is particularly challenging in large-scale or dynamic business environments (Leemans, Fahland, & van der Aalst, 2013).

4.5.5 DC5: User Adoption and Trust

Gaining user trust in automatically generated models is another key challenge. Business process experts accustomed to manual modeling may be skeptical of fully automated solutions, particularly regarding the accuracy and completeness of the generated models. To address this, the framework must prioritize transparency in the model generation process, allowing users to understand how the models are created and validated. Additionally, the framework should offer interfaces for user validation and feedback, enabling experts to review, adjust, and refine the models as necessary. By fostering trust through transparency and control, the framework can enhance its adoption among business process professionals (Busch et al., 2023).

4.5.1 DC6: Cost Management in API Use vs. Local Deployment

Managing the costs associated with using external services versus local deployment presents a significant challenge for the framework, especially in enterprise environments where large volumes of text must be processed regularly. External services, such as those offered by advanced model providers, come with recurring costs that can escalate quickly as usage increases. Conversely, deploying a local instance requires substantial upfront investment in computational resources, including specialized hardware, as well as ongoing maintenance expenses. Addressing this challenge necessitates a careful balance between cost, performance, and scalability. The framework could, for example, leverage external services for high-demand tasks while utilizing local resources for less intensive operations. Additionally, offering flexible deployment options would allow organizations to choose the approach that best aligns with their budget and performance needs (Touvron et al., 2023; Tomo Licardo et al., 2023).

5. PROOF-OF-CONCEPT SYSTEM AND EVALUATION

This section outlines the implementation of the framework and presents an evaluation using a series of test examples. The system, built in Python, leverages OpenAI’s GPT-4o models to automatically generate business process models from textual descriptions. This implementation is hosted on GitHub (https://github.com/LeonDragon/sProGen), and it features a web-based user interface developed using Flask. Users can input textual descriptions of processes, which the system processes to generate graphical models using Graphviz’s DOT language. The evaluation demonstrates the system’s capabilities through several examples.

5.1 System Implementation

The framework employs a structured approach consisting of two main stages: Automatic Process Model Generation and

Automatic Structural Verification of Business Processes. These stages utilize systematic prompts and large language models to transform unstructured text into structured business process models. The web-based user interface (UI) allows users to input textual narratives describing business processes, which the system then parses to identify entities, control flows, gateways, loops, and other components, generating a structured model. The resulting model is visualized as a graph using the DOT language and can be exported for integration with business process management (BPM) tools. An example of the UI is shown in Fig. 4.

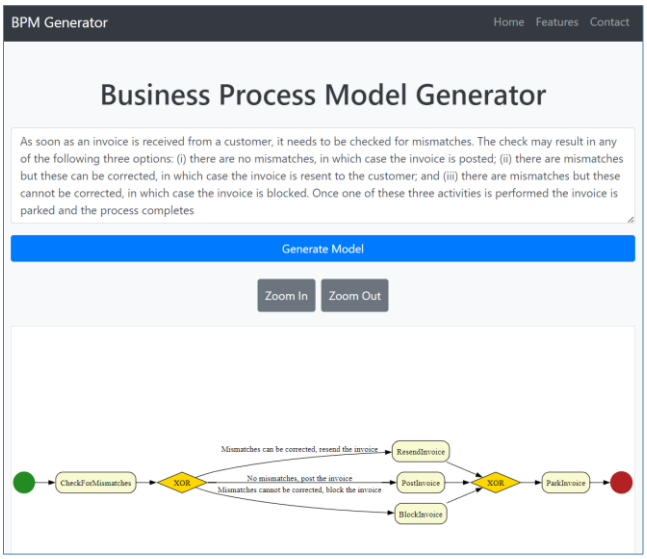

Fig. 4. User Interface of the Business Process Model Generator

5.2. Evaluation

The paper evaluates the efficacy and accuracy of the framework in generating process models from textual descriptions of business processes, as defined in Definition 1. The evaluation focuses on the system's ability to accurately interpret process information, manage concurrency and decision points, and translate them into coherent process models. We assess the outputs based on syntactic correctness (adherence to defined syntax rules), semantic correctness (accurate identification of process elements and relationships), and pragmatic correctness (usability and visual coherence). Our evaluation strategy tests the system with examples of varying complexity, starting with a textual process description followed by an analysis of the generated model. The results highlight the system's performance and identify any syntactical or structural issues.

Table 2. Examples of process descriptions from (Dumas, La Rosa, Mendling, & Reijers, 2018)

No	Process Name	Process description
1	Invoice checking	As soon as an invoice is received from a customer, it needs to be checked for mismatches. The check may result in any of the following three options: (i) there are no mismatches, in which case the invoice is posted; (ii) there are mismatches but these can be corrected, in which case the invoice is resent to the customer; and (iii) there are mismatches but these cannot be corrected, in which case the invoice is blocked. Once one of these three activities is performed the invoice is parked and the process completes.
2	Order-to-cash	Order-to-cash process starts whenever a purchase order has been received from a customer. The first activity that is carried out is confirming the order. Next, the shipment address is received so that the product can be shipped to the customer. Afterwards, the invoice is emitted and once the payment is received the order is archived, thus completing the process. Please note that a purchase order is only confirmed if the product is in stock, otherwise the process completes by rejecting the order. If the order is confirmed, the shipment address is received and the requested product is shipped while the invoice is emitted and the payment is received. Afterwards, the order is archived and the process completes.
3	Assessing loan applications	Once a loan application is received by the loan provider, and before proceeding with its assessment, the application itself needs to be checked for completeness. If the application is incomplete, it is returned to the applicant, so that they can fill out the missing information and send it back to the loan provider. This process is repeated until the application is found complete

Example 1 - Invoice checking: This process presents a straightforward business process with a basic sequential flow and a conditional decision.

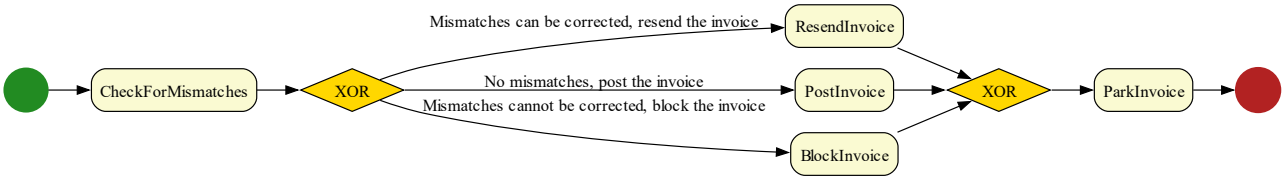

Fig. 5. Generated business process for the Invoice checking process description

Table 3. Generating and Evaluating the Invoice Checking Process

Key Elements	○	*Sequential flow*: CheckForMismatches → PostInvoice / ResendInvoice / BlockInvoice → ParkInvoice
	○	*Conditional decision*: The XOR gateway directs the flow based on the outcome of the mismatch check
Generated Model	○	The model generated (as shown in Fig. 5) by the system correctly identifies the sequence of tasks and the decision point. The XOR gateway is used to handle the conditional branch, ensuring that the process flows according to the description
Evaluation	○	*Syntactical Accuracy*: The system accurately models the sequence and decision logic, with no syntactical errors observed.
	○	*Soundness*: The model is sound
	○	*Comments*: The process flow is represented clearly, and the system handles simple decision points effectively

Example 2 - Order-to-Cash: The process demonstrates moderate complexity by incorporating both parallel flows and conditional decisions. The parallelism in handling the shipping and invoicing tasks increases the need for accurate synchronization, while the conditional decision based on product availability introduces branching logic. In general, the system effectively captures these complexities, ensuring that the concurrent activities and decision points are modeled accurately and efficiently.

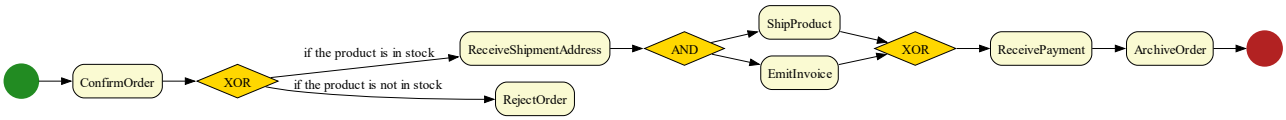

Fig. 6. Generated business process for the Order-to-Cash process description

Table 4. Generating and Evaluating the Order-to-Cash Process

Key Elements	○	*Sequential flow*: ConfirmOrder → ReceiveShipmentAddress → ReceivePayment → ArchiveOrder
	○	*Conditional decision*: The XOR gateway directs the flow based on product availability, either confirming or rejecting the order.
	○	*Parallel flow*: The AND gateway enables concurrent execution of ShipProduct and EmitInvoice activities.
Generated Model	○	The model generated (as shown in Fig. 6) ccurately represents the sequence of tasks, decision points, and parallel flows. The XOR gateway correctly handles the conditional logic, while the AND gateway ensures parallelism in shipping and invoicing. However, the convergence of AND split (ShipProduct and EmitInvoice) are not handle properly.
Evaluation	○	*Syntactical Accuracy*: The system accurately models the sequence and decision logic. However, the end event after RejectOrder is not properly illustrated in the generated process model.
	○	*Soundness*: The process model contains a lack of synchronization error at the XOR join.
	○	*Comments*: While the system effectively models the sequence and decision logic, there are issues with properly illustrating the end event after RejectOrder. Additionally, the process model encounters a lack of synchronization at the XOR join, indicating the need for further refinement to ensure soundness.

To address the issues identified during evaluation, the system leverages automated feedback loops that iteratively refine the process model. These feedback loops focus on resolving syntactical inaccuracies, such as properly illustrating the end event, and addressing soundness issues, such as the lack of synchronization at the XOR join. By continuously reprocessing and adjusting the model based on feedback, the system ensures that the generated process models are both accurate and logically sound, as shown in the updated diagram in Fig. 7.

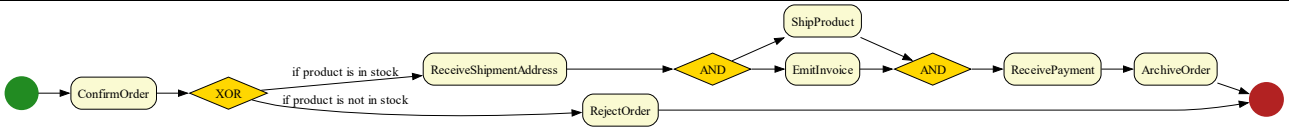

Fig. 7. Generated business process for the Order-to-Cash process description after feedback loop

Example 3 - Assessing loan applications: The process demonstrates moderate complexity by incorporating both conditional decisions and looping structures. The conditional decision at the completeness check introduces branching logic, while the looping mechanism ensures that incomplete applications are handled iteratively until they meet the required criteria. The system effectively captures these complexities, ensuring that decision points and the loopback logic are modeled accurately and efficiently.

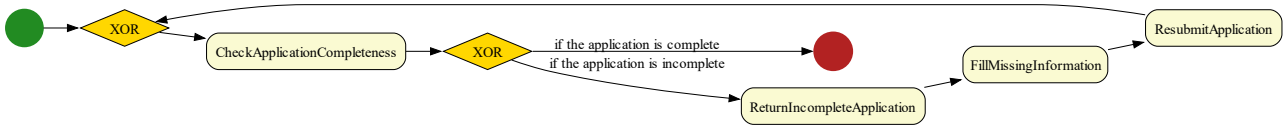

Fig. 8. Generated business process for the Assessing loan applications process description

Table 5. Generating and Evaluating the Order-to-Cash Process

Key Elements	○	*Sequential flow*: ReceiveLoanApplication → CheckApplicationCompleteness → AssessCompleteApplication.
	○	*Conditional decision*: The XOR gateway directs the flow based on application completeness, either moving forward for assessment or returning the application to the applicant
	○	*Loopback flow:* The XOR gateway enables the looping of the process for incomplete applications until all required information is provided.
Generated Model	○	The model generated (as shown in Fig. 8) accurately represents the sequence of tasks, decision points, and loopback flows. The XOR gateway correctly handles the conditional logic, ensuring that incomplete applications are returned to the applicant and resubmitted for further processing.
Evaluation	○	*Syntactical Accuracy*: The system accurately models the sequence and decision logic.
	○	*Soundness*: The process does not contain any error.
	○	*Comments*: The process flow is represented clearly, and the system handles decision points effectively.

6. CONCLUSION AND FUTURE DIRECTIONS

This paper presented the framework for generating accurate and sound business process models from text, which leverages large language models to automate the generation of business process models from unstructured textual descriptions. By integrating natural language processing with systematic prompt engineering, complex process narratives can be transformed into structured models, ensuring soundness through automatic verification and iterative refinement. The framework's key contributions lie in its ability to reduce manual effort, improve model accuracy, and ensure sound process models, offering a scalable solution for business process modeling across different domains. The use of multiple LLMs for cost-efficiency and flexibility further strengthens the framework's potential in real-world applications.

Despite its innovative approach, the framework has certain limitations that warrant further attention. One major challenge is handling ambiguous or complex textual descriptions, which can lead to inaccuracies in the generated process models. Although feedback loops are implemented to address these issues, the framework's ability to resolve ambiguities remains limited and requires improvement. Additionally, scalability becomes a concern as the complexity of process descriptions increases, posing challenges in maintaining performance and model soundness for large-scale processes. Moreover, while automatic feedback mechanisms help detect and correct errors, certain complex scenarios, such as nested loops or intricate decision points, still require manual intervention. Finally, the evaluation of the framework is currently based on a limited dataset, necessitating broader testing to fully assess its generalizability and robustness.

Future work will focus on enhancing the framework by leveraging advancements in LLM techniques to better handle complex and ambiguous textual descriptions, improving context understanding, reducing errors, and minimizing human intervention. Scalability will be addressed through fine-tuning or domain-specific training of LLMs, enabling management of larger and more intricate process narratives. Additionally, incorporating reinforcement learning-based feedback will allow the system to learn from corrections over time, further refining model generation. Expanding evaluation across diverse datasets and industries will ensure the framework's generalizability, while integration with existing BPM tools will smooth organizational adoption. As LLM techniques evolve, the framework could extend to support predictive analytics, automated process optimization, and real-time process monitoring, positioning it at the forefront of automated business process modeling.

7. ACKNOWLEDGMENTS

The authors gratefully acknowledge the support and contributions of all colleagues and institutions involved in this research.

8. REFERENCES

Aalst, W. M. P. (1997). Verification of workflow nets. In P. Azéma & G. Balbo (Eds.), *Application and Theory of Petri Nets 1997* (pp. 407–426). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-63139-9_48

Bellan, P., Dragoni, M., & Ghidini, C. (2020). A qualitative analysis of the state of the art in process extraction from text. *Proceedings of the AIxIA 2020 Discussion Papers Workshop Co-Located with the 19th International Conference of the Italian Association for Artificial Intelligence (AIxIA2020)*, *2776*, 19–30. CEUR-WS.org.

Berti, A., Schuster, D., & van der Aalst, W. M. P. (2023). Abstractions, scenarios, and prompt definitions for process mining with LLMs: A case study. *Business Process Management Workshops - BPM 2023 International Workshops*, *492*, 427–439. Springer.

Busch, K., Rochlitzer, A., Sola, D., & Leopold, H. (2023). Just Tell Me: Prompt Engineering in Business Process Management. In *Lecture Notes in Business Information Processing*: *Vol. 479 LNBIP* (pp. 3–11). https://doi.org/10.1007/978-3-031-34241-7_1

Chen, S., & Liao, H. (2022). Bert-log: Anomaly detection for system logs based on pre-trained language model. *Applied Artificial Intelligence*, *36*(1).

Choi, Y., Kongsuwan, P., Joo, C. M., & Zhao, J. L. (2015). Stepwise structural verification of cyclic workflow models with acyclic decomposition and reduction of loops. *Data & Knowledge Engineering*, *95*, 39–65. https://doi.org/10.1016/j.datak.2014.11.003

de A. R. Gonçalves, J. C., Santoro, F. M., & Baião, F. A. (2011). Let me tell you a story - on how to build process models. *Journal of Universal Computer Science*, *17*(2), 276–295.

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. A. (2018). *Fundamentals of Business Process Management*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-56509-4

Fahland, D., Favre, C., Koehler, J., Lohmann, N., Völzer, H., & Wolf, K. (2011). Analysis on demand: Instantaneous soundness checking of industrial business process models. *Data and Knowledge Engineering*, *70*(5), 448–466. https://doi.org/10.1016/j.datak.2011.01.004

Forell, M., & Schüler, S. (2024). *Modeling meets Large Language Models*. https://doi.org/10.18420/modellierung2024-ws-003

Friedrich, F., Mendling, J., & Puhlmann, F. (2011a). Process model generation from natural language text. *Advanced Information Systems Engineering - 23rd International Conference, CAiSE 2011*, *6741*, 482–496. Springer.

Friedrich, F., Mendling, J., & Puhlmann, F. (2011b). Process Model Generation from Natural Language Text. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 6741, pp. 482–496). https://doi.org/10.1007/978-3-642-21640-4_36

Grohs, M., Abb, L., Elsayed, N., & Rehse, J.-R. (2024). Large Language Models Can Accomplish Business Process Management Tasks. In *Business Process Management Workshops - BPM 2023 International Workshops* (Vol. 492, pp. 453–465). Springer. https://doi.org/10.1007/978-3-031-50974-2_34

Hans-Georg, F., Peter, F., & Julius, K. (2023). Conceptual Modeling and Large Language Models: Impressions From First ExperimentsWith ChatGPT. *Enterprise Modelling and Information Systems Architectures*, *18*(January). https://doi.org/10.18417/emisa.18.3

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75. https://doi.org/10.2307/25148625

Ivanchikj, A., Serbout, S., & Pautasso, C. (2020). From text to visual BPMN process models: design and evaluation. *MoDELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems*, 229–239. ACM.

Kampik, T., Warmuth, C., Rebmann, A., Agam, R., Egger, L. N. P., Gerber, A., … Weidlich, M. (2024). Large Process Models: A Vision for Business Process Management in the Age of Generative AI. *KI - Künstliche Intelligenz*, (0123456789). https://doi.org/10.1007/s13218-024-00863-8

Klievtsova, N., Benzin, J.-V., Kampik, T., Mangler, J., & Rinderle-Ma, S. (2023). Conversational process modelling: State of the art, applications, and implications in practice. *Business Process Management Forum - BPM 2023 Forum*, *490*, 319–336. Springer.

Kourani, H., Berti, A., Schuster, D., & van der Aalst, W. M. P. (2024). Process Modeling with Large Language Models. In H. van der Aa, D. Bork, R. Schmidt, & A. Sturm (Eds.), *Enterprise, Business-Process and Information Systems Modeling* (pp. 229–244). Cham: Springer Nature Switzerland.

Leemans, S. J. J. J., Fahland, D., & van der Aalst, W. M. P. P. (2013). Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. In J.-M. Colom & J. Desel (Eds.), *Application and Theory of Petri Nets and Concurrency: 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings*: *Vol. 7927 LNCS* (pp. 311–329). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38697-8_17

Muff, F., & Fill, H.-G. (2024). Limitations of ChatGPT in Conceptual Modeling: Insights from Experiments in Metamodeling. *Modellierung*. Retrieved from https://api.semanticscholar.org/CorpusID:269529265

OMG. (2013). Object Management Group, Business process modeling notation (BPMN) version 2.0.2. In *Object Management Group*. https://doi.org/10.1007/978-3-642-33155-8

OpenAI. (2022). Introducing ChatGPT. Retrieved December 8, 2024, from https://openai.com/index/chatgpt/

Ouyang, C., Adams, M., ter Hofstede, A. H. M., & Yu, Y. (2018). *Towards the Design of a Scalable Business Process Management System Architecture in the Cloud*. https://doi.org/10.1007/978-3-030-00847-5_24

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Prinz, T. M., Prinz, T., & De, W. A. (2017). *Why We Need Static Analyses of Service Compositions - Fault vs . Error Analysis of Soundness*.

Sholiq, S., Sarno, R., & Astuti, E. S. (2022). Generating BPMN diagram from textual requirements. *Journal of King Saud University - Computer and Information Sciences*, *34*(10), 10079–10093. https://doi.org/10.1016/j.jksuci.2022.10.007

Tomo Licardo, J., Tanković, N., & Etinger, D. (2023). *A Method for Extracting BPMN Models from Textual Descriptions Using Natural Language Processing*. Retrieved from https://urn.nsk.hr/urn:nbn:hr:137:105792

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., … Lample, G. (2023). *LLaMA: Open and Efficient Foundation Language Models*. Retrieved from https://arxiv.org/abs/2302.13971

van der Aalst, W. M. P., Hirnschall, A., & Verbeek, H. M. W. (2002). An Alternative Way to Analyze Workflow Graphs. In *CAiSE'02* (pp. 535–552). https://doi.org/10.1007/3-540-47961-9_37

Vidgof, M., Bachhofner, S., & Mendling, J. (2023). *Large Language Models for Business Process Management: Opportunities and Challenges*. https://doi.org/10.1007/978-3-031-41623-1_7