Beyond Control Flow: Integrating Organizational Perspectives into Generative Process Modeling

Anton Antonov[1] , **Humam Kourani**[1] , **Alessandro Berti**[2]
[1]Fraunhofer Institute for Applied Information Technology FIT, Schloss Birlinghoven, 53757 Sankt Augustin, Germany
[2]RWTH Aachen University, Ahornstraße 55, 52074 Aachen, Germany
{humam.kourani,anton.antonov}@fit.fraunhofer.de, a.berti@pads.rwth-aachen.de

Abstract

Process modeling is essential for analyzing and optimizing organizational workflows, encompassing multiple dimensions such as control flow, resources, data, and the operational perspective. However, traditional modeling approaches often require extensive event data or specialized domain expertise, limiting their accessibility to non-experts. While recent approaches utilizing Generative AI (GenAI) have simplified the generation of process models from text, they predominantly focus on the control-flow perspective. Consequently, existing approaches neglect the resource perspective, which is critical for identifying organizational structures and accountability. Integrating this perspective is non-trivial because direct generation of standard notations, e.g., the industry standard *Business Process Model and Notation* (BPMN), is error-prone and difficult to validate, while existing intermediate representations lack native support for the BPMN constructs used to distinguish organizations and roles. Additionally, the automated visualization of such multi-perspective models remains a technical challenge due to the lack of BPMN auto-layout algorithms that support the organizational dimension. In this paper, we introduce a robust modeling approach capable of generating multi-perspective process models from natural language text using Large Language Models (LLMs). We achieve this by defining an extended intermediate representation that explicitly captures resource elements, which is subsequently transformed into syntactically correct BPMN diagrams with auto-generated layout. We evaluate our approach across nine LLMs. The results confirm high accuracy in role discovery without compromising control-flow quality or computational efficiency, demonstrating that comprehensive multi-perspective process modeling is achievable with GenAI.

1 Introduction

The field of process modeling encompasses techniques for the representation and analysis of business processes. Comprehensive process models typically address several key perspectives [Dumas *et al.*, 2018]: the *control-flow perspective*, which dictates the ordering of these activities and events; the *data perspective*, denoting entities that flow through activities; the *operational perspective*, which defines the process execution rules; and the *resource (or organizational) perspective*, which specifies the actors and roles involved in the process execution. Process modeling allows for the formal analysis of workflows and relies on the industry standard *Business Process Model and Notation (BPMN) 2.0* [OMG, 2011]. This notation captures the control flow via *tasks*, *gateways*, and *sequence flows*; data via *data objects*; and resources via *pools* that model organizational entities, *lanes* that model roles within organizations, and *message flows* that define communication between distinct organizational entities.

Standard modeling languages are often syntactically complex, creating a high barrier to entry for non-experts [Recker, 2010; Rosemann, 2006]. Consequently, Large Language Models (LLMs) have emerged as an alternative to manual modeling approaches. By leveraging their extensive pretrained knowledge and ability to interpret unstructured textual inputs, LLMs can significantly streamline the modeling process for non-technical users [Kourani *et al.*, 2025b; Licardo *et al.*, 2025; Köpke and Safan, 2024]. Current approaches in generative process modeling focus predominantly on integrating the control-flow perspective, resulting in models that are functionally accurate but organizationally agnostic. By prioritizing the sequencing of events, these methods overlook the organizational perspective, which describes the actors and roles associated with those events. Without capturing the interplay between different organizations or roles, such models provide only a partial view of the process.

Integrating the resource perspective into generative process modeling is not a trivial task. First, the direct generation of verbose standard formats, such as BPMN 2.0 XML, forces the LLM to manage strict syntactical constraints and complex graph topologies alongside process logic. This overhead makes the generation highly prone to errors and necessitates extensive schema definitions, a practice shown to degrade the model's reasoning capabilities and output quality [Wang *et al.*, 2024]. Furthermore, automatically validating the structural and semantic quality of raw XML outputs is exceptionally difficult. In contrast, utilizing a compact intermediate representation abstracts away serialization complexity, sim-

plifying the generation task and enabling robust, automated validation mechanisms. Finally, the automated visualization of complex BPMN models presents a distinct technical challenge. While theoretical layout approaches exist [Kitzmann *et al.*, 2009], current tooling lacks robust automated support for pools and lanes [Berti *et al.*, 2023], necessitating extensive manual layouting.

In this paper, we present a method integrating the organizational perspective into generative process modeling, building upon the ProMoAI framework [Kourani *et al.*, 2024a; Kourani *et al.*, 2024b]. Our reliance on its intermediate representation is a deliberate design choice; recent findings in grammar-guided generation demonstrate that projecting complex tasks into constrained languages significantly reduces hallucination rates by narrowing the output search space [Wang *et al.*, 2023]. Therefore, we first extend the underlying intermediate representation to explicitly support resources. Second, we define a deterministic transformation that maps this extended representation to standard BPMN 2.0. Third, we address the visualization challenge by implementing an automated layout algorithm tailored for resource-aware BPMN models.

To illustrate the critical importance of the resource perspective, consider the *Complaint Handling* process depicted in Figure 1. Figure 1 (a) displays the model generated by the baseline ProMoAI using *gemini-3-flash*. While this model correctly captures the logical sequence of the control flow, it presents a "flat" view of the process. Crucially, it fails to indicate that the *Finance* department is specifically responsible for reimbursement, distinct from the *Customer Service* team handling the dispute. From a compliance and analysis standpoint, this representation is incomplete. In contrast, Figure 1 (b) demonstrates the output of our proposed method. By explicitly modeling resources, the system generates a full collaboration diagram containing pools, lanes, and message flows that define the communication between the *Customer* and the *Company*. This enrichment transforms the model from a simple sequence of steps into a comprehensive blueprint of organizational interaction.

The rest of the paper is organized as follows. Section 2 reviews related work on generative process modeling and multi-perspective representations. Section 3 summarizes the ProMoAI framework and its intermediate representation. Section 4 then presents our resource-aware extension, including the extended modeling language, the deterministic transformation to BPMN, and the automated layout algorithm. Section 5 reports the experimental setup and evaluation results across nine LLMs. Finally, Section 6 concludes the paper and outlines directions for future work.

2 Related Work

This work builds upon ProMoAI [Kourani *et al.*, 2024b; Kourani *et al.*, 2024a], which serves as an ideal foundation due to its extensible architecture and its reliance on an intermediate representation. This approach guarantees structural correctness and soundness by design, ensuring that the generated control flow remains valid even as the model complexity increases. We enhance the underlying intermediate representation used for LLM prompting to explicitly support organizational constructs, and we define a deterministic transformation from this representation to standard BPMN. While other approaches leveraging GenAI for process modeling exist, they suffer from distinct limitations. Some require resource-intensive model fine-tuning [Nivon and Salaün, 2024], while others rely on generic data formats, such as JSON [Köpke and Safan, 2024; Licardo *et al.*, 2025]. However, as a generic format, JSON is not optimized for process semantics. It struggles to naturally express complex behavioral patterns, particularly when handling loops of choices or nested gateways, resulting in representations that are difficult for LLMs to generate consistently. Taking a different approach, [Lin *et al.*, 2024] propose a multi-agent framework designed to emulate the iterative stages of manual process modeling. While this method achieves high accuracy by simulating human reasoning, its complex multi-agent architecture introduces significant computational overhead. Nevertheless, the complexity of this multi-agent framework makes the inclusion of a human-in-the-loop infeasible. Injecting user feedback into such a convoluted automated workflow is non-trivial, limiting the system's adaptability.

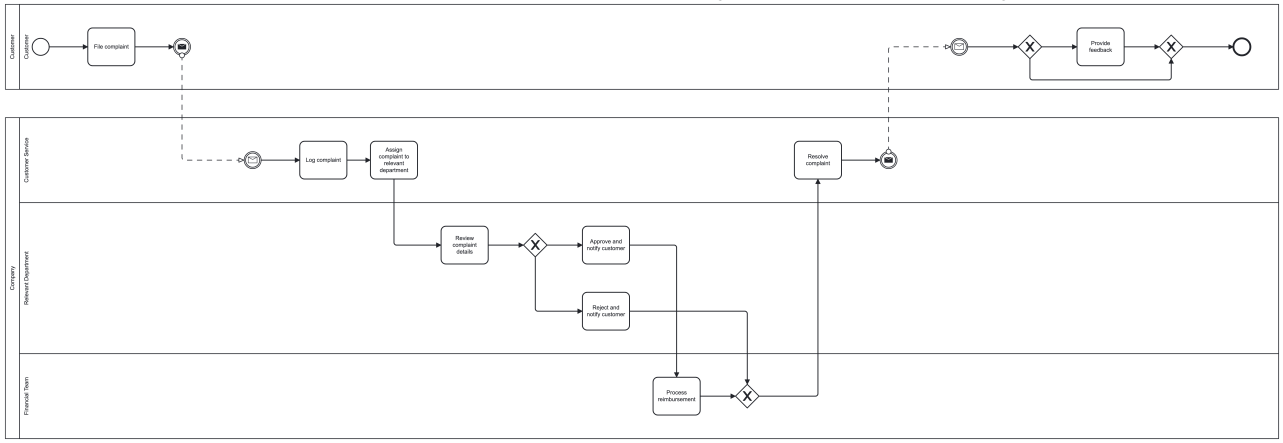
A limited number of studies have addressed multi-perspective modeling. For instance, [Klievtsova *et al.*, 2025] discuss a method for integrating the data perspective. However, their implementation relies on internal structures and transformation steps that are not fully disclosed. Additionally, their approach produces models that depart from the standard BPMN specification, reducing their compatibility with existing process management tools. While [Safan and Köpke, 2025] extend current approaches to include both the resource perspective and the data perspective, their contribution prioritizes engineering specifics over a clear definition of the underlying method, leaving several transformation steps not discussed.

3 Overview of ProMoAI

ProMoAI serves as a comprehensive framework for the generation and iterative refinement of process models based on natural language text. As illustrated in Figure 2, ProMoAI's architecture [Kourani *et al.*, 2024b] comprises several components: low-level intermediate representation, prompting module, process generation, error handling, and user refinement. We briefly describe them.

Intermediate Representation Language. ProMoAI employs a Python-based low-level language as an intermediate representation to generate hierarchical process models based on the Partially Ordered Workflow Language (POWL) [Kourani and van Zelst, 2023]. This design choice leverages two key advantages. First, the hierarchical nature of POWL allows for the recursive composition of complex process structures from simple blocks, ensuring that the generated models remain modular and sound by construction. Second, by representing this logic in Python code, the framework exploits the state-of-the-art code generation capabilities of modern LLMs [Bistarelli *et al.*, 2025; Zan *et al.*, 2022; Jiang *et al.*, 2024], which are more robust than their ability to generate raw XML or graph structures directly. The

(a) Baseline generation (ProMoAI): The model captures the logical sequence using standard control-flow elements: *tasks*, *start/end events*, *gateways*, and *sequence flows*. However, the organizational context is missing.



(b) Our approach: The model integrates the organizational perspective. In addition to control flow, it utilizes *pools* and *lanes* to define roles, and *message flows* with *message events* to represent inter-organizational communication.

Figure 1: Motivating Example: Models generated by *gemini-3-flash* for the *Complaint Handling* process (p13).

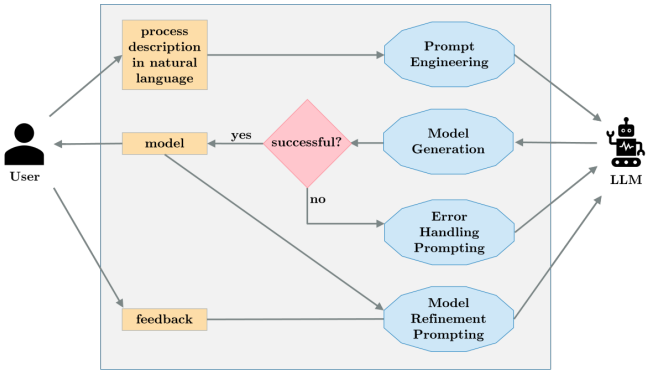


Figure 2: Architecture of ProMoAI.

expressions defined in the intermediate language are transformed into the formal POWL model using a dedicated semantic mapping function.

Prompt Engineering ProMoAI employs a structured prompt engineering method to guide the LLM into constructing process models. The pipeline begins with role prompting [Xu *et al.*, 2023], used to introduce the subsequent task. Afterwards, detailed information about the intermediate language is provided using *least-to-most prompting* [Zhou *et al.*, 2023]. Such an approach instructs the model to generate the process hierarchically, defining atomic structures before composing them into higher-order control structures. Additionally, the prompt includes a curated set of few-shot positive and negative examples to demonstrate correct usage and common mistakes [Brown *et al.*, 2020; Miyake *et al.*, 2025].

Process Generation The LLM is instructed to output executable Python code. Prior to execution, the code undergoes static validation to verify adherence to the used syntax and to detect potential security risks, such as the use of unauthorized libraries.

Error Handling ProMoAI incorporates a self-correcting error handling strategy based on iterative prompting [Jha *et al.*, 2023]. Whenever a syntax error or execution failure is detected, the system captures the specific exception and feeds it back into the conversation context as an informative prompt. This allows the LLM to analyze the error and generate a corrected version of the code in subsequent turns.

Human-in-the-Loop Feedback Furthermore, the framework supports a Human-in-the-Loop workflow. End users can provide natural language feedback to refine the generated models, which is injected into the conversation context, enabling the LLM to iteratively adjust the process models.

4 Integrating the Resource Perspective

In this section, we detail how we extend ProMoAI to support the resource perspective.

4.1 Process Model Generation

Low-Level Process Modeling Language

To incorporate the organizational perspective, we extend the intermediate language's syntax to natively support resource definitions. Specifically, we modify the atomic activity constructor, *gen.activity*, to accept two additional mandatory arguments: *pool* and *lane*. This integration represents a deliberate design choice to enforce a single-stage generation process. By requiring the LLM to define control flow and resources simultaneously, we avoid the inconsistencies and latency asso-

```python
cust = {"pool": "Customer", "lane": "Customer"}
cs   = {"pool": "Company",  "lane": "Cust. Service"}
dept = {"pool": "Company",  "lane": "Rel. Dept."}
fin  = {"pool": "Company",  "lane": "Finance"}

file    = gen.activity("File complaint", **cust)
feed    = gen.activity("Provide feedback", **cust)
log     = gen.activity("Log complaint", **cs)
assign  = gen.activity("Assign dept", **cs)
review  = gen.activity("Review details", **dept)
appr    = gen.activity("Approve & notify", **dept)
rej     = gen.activity("Reject & notify", **dept)
pay     = gen.activity("Reimburse", **fin)
resolve = gen.activity("Resolve case", **cs)

model = gen.decision_graph([
    (None, file), (file, log), (log, assign), (assign,
    ↪  review),
    (review, appr), (review, rej),
    (appr, pay), (pay, resolve),
    (rej, resolve),
    (resolve, feed), (resolve, None),
    (feed, None)
])
```

Figure 3: Intermediate representation generated using *gemini-3-flash* for the Complaint Handling process (p13).

ciated with multi-stage pipelines (e.g., generating the process structure first and post-annotating roles). The definitions of the recursive control-flow structures remain invariant, allowing us to leverage the existing soundness guarantees of the original POWL framework. An example of the extended syntax is provided in Figure 3.

Prompt Engineering

Due to the necessary changes in the intermediate language representation, we revise the prompt engineering strategy outlined in ProMoAI. Specifically, we augment the system instructions to explicitly define the concepts of pools and lanes, ensuring the LLM understands the organizational context. Furthermore, we update the few-shot examples accordingly.

Model Generation and Error Handling

We extend the existing model validation mechanism to enforce semantic consistency in resource assignment. A common failure mode in LLM generation involves partial resource attribution, where the model assigns pools and lanes to a subset of activities while leaving others undefined. We treat this as a validation error, based on the principle that if organizational information is discoverable for any part of the process, it should be consistently applied to all relevant tasks.

4.2 Generating Resource-Aware BPMN Models

A key challenge in automated BPMN generation is ensuring the resulting diagrams are visually coherent [Abbad-Andaloussi *et al.*, 2025; Lübke *et al.*, 2021]. While automatic layout algorithms exist for simple control-flow models, the effective placement and routing of elements within pools and lanes remains an open challenge.

Transforming POWL into BPMN follows a recursive approach, leveraging the fact that POWL models are inherently recursive. In the original pipeline [Kourani *et al.*, 2025b], this conversion relied on an intermediate Petri net representation. However, standard Petri nets do not natively support organizational constructs (pools, lanes, message events, and message flows). To address this, we introduce a direct translation pipeline (POWL $\rightarrow$ BPMN) that constructs and layouts the model in three sequential phases:

- **Structural Construction and Consolidation:** The algorithm traverses the POWL model bottom-up. For each atomic activity, it generates a BPMN task explicitly assigned to the pool and lane defined in the intermediate representation. For complex POWL constructs (e.g., choice graphs), it uses BPMN gateways and recursively assembles the child subgraphs.

  During this phase, we apply a *structural consolidation* heuristic: contiguous control-flow blocks (e.g., paired XOR-splits and joins) are identified and logically grouped within the same organizational boundary where possible. This minimizes cross-pool message flows and ensures that logical units of work are visually clustered together.

- **Context-Aware Link Generation:** After all BPMN flow nodes have been created and assigned to pools/lanes, we generate edges by translating each precedence relation (u, v) induced by the POWL model into BPMN-compliant connectors. Let $\mathsf{pool}(\cdot)$ and $\mathsf{lane}(\cdot)$ denote the organizational assignment of a node. If $\mathsf{pool}(u) = \mathsf{pool}(v)$, we add a BPMN *sequence flow* from the BPMN element corresponding to u to that of v (within the same pool). If $\mathsf{pool}(u) \neq \mathsf{pool}(v)$, we *never* create a sequence flow across pools; instead, we create a *message interaction* by inserting (i) a message-throwing event $e^{out}_{u,v}$ in the source pool $\mathsf{pool}(u)$ (placed in $\mathsf{lane}(u)$) and (ii) a message-catching event $e^{in}_{u,v}$ in the target pool $\mathsf{pool}(v)$ (placed in $\mathsf{lane}(v)$). We then add sequence flows $u \rightarrow e^{out}_{u,v}$ and $e^{in}_{u,v} \rightarrow v$, and a BPMN *message flow* $e^{out}_{u,v} \dashrightarrow e^{in}_{u,v}$ connecting the two pools. To satisfy BPMN connection constraints, message flows are attached only to the inserted message events (or tasks), never directly to gateways: if u (resp. v) is a gateway, the event $e^{out}_{u,v}$ (resp. $e^{in}_{u,v}$) is inserted immediately on that specific outgoing (resp. incoming) branch and linked to the gateway via a sequence flow. Finally, when a cross-pool message is the first trigger of the receiving pool (i.e., v has no in-pool predecessor), we instantiate $e^{in}_{u,v}$ as a *message start event*; analogously, when a cross-pool message is emitted as the last step of the sending pool (i.e., u has no in-pool successor), we instantiate $e^{out}_{u,v}$ as a *message end event*. This yields a well-formed BPMN collaboration diagram in which intra-organizational causality is expressed via sequence flows and inter-organizational causality via message flows with explicit send/receive semantics.

- **Orthogonal Layout and Routing:** Finally, to generate a readable diagram with minimized edge crossings and clear routing paths, we employ an orthogonal graph layout algorithm based on the Manhattan metric [Sander, 1995]. This ensures that sequence and message flows

are routed using horizontal and vertical segments, optimizing the spatial arrangement of pools and lanes.

5 Evaluation

In this section, we evaluate our approach against the baseline ProMoAI using a diverse set of state-of-the-art LLMs. The aim of our evaluation is (i) to validate the effect of the added task complexity on the time performance and control-flow quality, and (ii) to assess the quality of the added resource elements by comparing them against human-verified ground truth.

5.1 Experimental Settings

We evaluate our approach on a dataset of ten diverse business processes selected from the benchmark presented in [Kourani *et al.*, 2025b]. We filtered the original dataset to retain only processes that contain sufficient detail regarding process participants. Since the original ground truth BPMNs were limited to control flow, we manually annotated them with pools and lanes. This curation was necessary as existing large repositories of process models (e.g., [Corradini *et al.*, 2021; Sola *et al.*, 2022]) either lack paired textual descriptions or the ground-truth organizational layers necessary to evaluate multi-perspective modeling. Table 1 summarizes the characteristics of the selected processes.

To demonstrate the model-agnostic nature of our approach, we perform the evaluation using a diverse set of nine LLMs. This selection includes proprietary state-of-the-art models (e.g., from OpenAI, Anthropic, and Google) as well as open-source models.

5.2 Evaluation Metrics

(i) Effect of Added Task Complexity

A primary concern in extending generative modeling is whether the increased cognitive load (requiring the LLM to manage pools and lanes simultaneously with control flow) negatively impacts the generation process. We compare our resource-aware approach against the baseline (ProMoAI without resources) using the following metrics:

Control-flow Quality Score: We report a control-flow quality score, computed as described in [Kourani *et al.*, 2025b]. This metric quantifies how well the behavioral semantics of the generated model align with the ground-truth event logs. A comparable score between the baseline and our method would indicate that adding resource constraints does not disrupt the logic of the process.

Number of Needed Iterations: Since our pipeline includes an automated self-correction mechanism, we track the number of iterations required to produce a syntactically valid and executable model. An increase in iterations would suggest that the model struggles to adhere to the extended syntax or logical constraints.

Time per Iteration: Beyond the number of errors, we analyze the execution time per generation step. This metric helps distinguish whether the added complexity forces the LLM to spend more time "thinking" or generating longer token sequences.

(ii) Qualitative Assessment for the Resource Perspective

To validate the organizational perspective, we assess how accurately the LLMs identify resource elements and assign them to the process activities. As exact string matching is insufficient due to linguistic variability, we compare the generated resource elements against the extended ground truth using two complementary scores:

Semantic Similarity Score: To account for valid paraphrases and abbreviations (e.g., "HR" vs. "Human Resources"), we compute a semantic similarity score between generated and ground-truth resource labels using a pre-trained sentence embedding model [Reimers and Gurevych, 2021] (`all-MiniLM-L6-v2`). Since resources are attached to activities, we align predictions to ground truth *by activity name*: for every activity a that appears in both mappings, we embed the generated and ground-truth pool labels, $pool_{gen}(a)$ and $pool_{gt}(a)$, and compute their cosine similarity; we repeat the same procedure for lane labels, $lane_{gen}(a)$ and $lane_{gt}(a)$. If a pool/lane is unspecified, we represent it as `None` and include it in the comparison. The resulting per-activity similarities are averaged to obtain a per-process *Pool Similarity* and *Lane Similarity*, which we then aggregate over all processes to report model-level results. Reporting pools and lanes separately allows us to disentangle coarse-grained actor identification (pools) from fine-grained role/department naming (lanes).

Judge-based Quality Score: We implement an independent LLM-as-a-Judge protocol [Ji *et al.*, 2023] with a fixed evaluator model (*Grok-4.1 Fast Reasoning*) that is separate from the LLMs used for generation. For each generated process instance, the judge receives (i) the original textual description and (ii) the list of activities together with their predicted `pool`/`lane` assignments, but it is *not* provided with the ground-truth labels. The judge assigns a graded correctness score $s(a) \in [0, 1]$ for every activity a, reflecting whether the proposed organizational assignment is supported by the textual evidence (minor naming variations such as abbreviations or paraphrases are accepted; if the description provides no basis to assign a resource, `None` is considered a valid assignment). The judge is instructed to return a strict JSON object of the form $\{activity \mapsto score\}$; we automatically validate the response format and completeness, and re-prompt the judge (up to five attempts) only if the JSON is malformed or activities are missing. The judge-based score for a process is computed as the mean of the per-activity scores, and we aggregate these scores over all processes to report model-level results.

5.3 Evaluation Results

On the Task Complexity

Impact on Control-Flow Quality: Our experiments prompt that the control-flow quality scores for the resource-aware models closely mirror those of the baseline (cf. the aggregated results in Table 2). To rigorously validate this observation, we conduct significance testing on the control-flow quality scores using a paired t-test [Casella and Berger,

Table 1: Characteristics of the selected ground truth processes.

Process	Model Metrics				Structural Features		
	# Act.	# Gateways	# Pools	# Lanes	Decis.	Cycles	Concur.
Sales Order (p1)	8	6	2	4	✓		✓
Hiring Process (p2)	16	10	1	3	✓	✓	✓
Procurement Process (p5)	11	4	2	2	✓	✓	✓
Booking System (p7)	13	20	2	3	✓	✓	✓
Incident Reporting (p8)	13	4	2	3	✓		✓
Prototype Building (p9)	13	8	1	2	✓	✓	✓
Subscription Service (p11)	12	12	2	3	✓	✓	✓
Complaint Handling (p13)	9	4	2	4	✓		
Internal Audit (p16)	24	14	1	5	✓	✓	✓
University Admission (p18)	26	22	2	6	✓	✓	✓

Table 2: Comparison of the control-flow quality scores (Mean ± SD) between the baseline (control-flow only) and our resource-aware approach. The *p*-values (paired t-test) indicate no statistically significant difference ($p > 0.05$) for any model.

Model	Control-Flow Quality		*p*-value
	Baseline	Ours	
Grok-4 Fast Reasoning	0.921 ± 0.072	$\mathbf{0.924} \pm 0.069$	0.704
DeepSeek-v3.2	$\mathbf{0.917} \pm 0.041$	0.895 ± 0.062	0.231
GPT 5.2	0.909 ± 0.075	$\mathbf{0.915} \pm 0.064$	0.690
Gemini-3 Flash	0.910 ± 0.072	$\mathbf{0.920} \pm 0.058$	0.452
Claude Sonnet 4.5	$\mathbf{0.916} \pm 0.070$	0.899 ± 0.069	0.522
Kimi K2	$\mathbf{0.895} \pm 0.067$	0.835 ± 0.118	0.103
Claude Haiku 4.5	0.876 ± 0.076	$\mathbf{0.902} \pm 0.066$	0.172
Qwen3 Next 80B	$\mathbf{0.876} \pm 0.073$	0.863 ± 0.093	0.575
GPT-5 Mini	$\mathbf{0.904} \pm 0.091$	0.882 ± 0.095	0.205

Table 3: Average number of iterations required per model (Mean ± Std. Dev.). Lower values indicate higher generation stability.

Model	Baseline	Ours
Claude Haiku 4.5	1.60 ± 1.29	1.60 ± 1.03
Claude Sonnet 4.5	1.00 ± 0.00	1.00 ± 0.00
DeepSeek-v3.2	1.27 ± 0.45	1.60 ± 0.92
GPT 5.2	1.00 ± 0.00	1.00 ± 0.00
GPT-5 Mini	1.20 ± 0.40	1.00 ± 0.00
Gemini-3 Flash	1.20 ± 0.40	1.30 ± 0.46
Grok-4 Fast Reasoning	1.00 ± 0.00	1.20 ± 0.40
Kimi K2	1.50 ± 0.50	$\mathbf{2.86 \pm 1.89}$
Qwen3 Next 80B	2.75 ± 1.54	2.57 ± 1.41

2024]. The analysis reveals that for all cases the difference in performance between the control-flow-only and resource-aware models is not statistically significant ($p > 0.05$). The obtained p-values range from 0.103 to 0.704, confirming that the additional complexity of generating pools and lanes does not degrade the underlying process logic.

Efficiency Analysis Table 3 includes statistics about the required number of generation iterations until a model is generated successfully. The vast majority of models exhibited remarkable stability, as the number of iterations remained virtually the same between both settings. For instance, Claude Sonnet 4.5 and GPT 5.2 achieved perfect generation scores (1.00 ± 0.00) in both settings. The only notable exception is Kimi K2 (highlighted in bold), where the average number of required iterations increased significantly from 1.50 ± 0.5 to 2.86 ± 1.89 . This performance degradation indicates that the model's reasoning capabilities may be less robust when handling increased cognitive load. Specifically, Kimi K2 appears to struggle with the multi-objective nature of the task, simultaneously managing the control-flow logic and the resource assignment constraints, leading to a higher frequency of syntactic or logical errors that require iterative correction.

Figure 4 compares the runtime distributions for the baseline (control-flow only) versus our resource-aware approach per iteration. Despite the increased length of the generated code, necessitated by the additional pool and lane arguments, we observe that the runtime impact is negligible for the majority of models. Fast reasoning models like Grok-4 and Claude Sonnet 4.5 exhibit virtually identical runtime profiles in both settings. A slight increase is observed in some larger models, e.g., GPT 5.2 (from $\approx$ 40 seconds to $\approx$ 60 seconds). This is expected due to the increased number of tokens needed for the integration of the resource perspective. Nevertheless, these values still remain in an acceptable limit for process modeling.

Assessment of the Resource Perspective

Next, we evaluate the semantic accuracy of the discovered pool and lane labels. Figure 5 details the result. We observe that the Judge-based accuracy (white bars) remains consistently high, with top-performing models such as GPT-5 Mini achieving near-perfect scores across all processes. This confirms that the discovered assignments are logically sound. However, notable deviations appear in the Pool Similarity (light blue bars). This discrepancy highlights the lexical heterogeneity inherent in process descriptions: while the model may assign a valid organizational entity (approved by

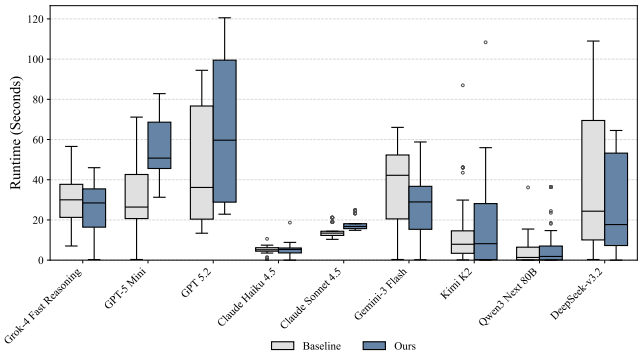


Figure 4: Boxplot analysis of execution time (seconds) per iteration. The consistent distributions between the baseline (grey) and our approach (blue) demonstrate the efficiency of the extended generation pipeline.

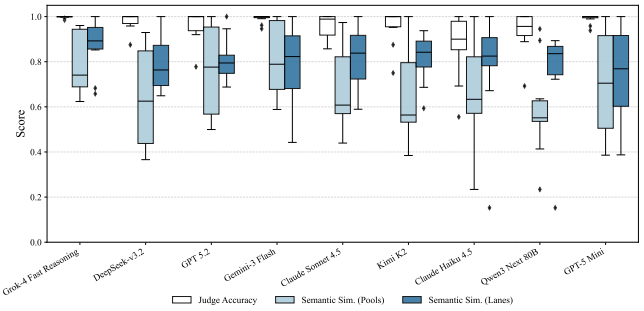


Figure 5: Accuracy of resource assignment aggregated by LLMs.

the Judge), its specific naming convention, e.g., "Corporate Headquarters" vs. "Central Audit Team" in p16, or "Product Development Organization" vs. "Organization" in p9, may differ vectorially from the ground truth. Furthermore, certain activities inherently span multiple organizational boundaries (e.g., the "Sign Contract" activity in p5, which involves both the Supplier and the Procuring Entity). In such cases, assigning either party is a valid modeling decision, yet it penalizes the similarity score if it mismatches the ground truth. Consequently, we interpret Semantic Similarity as a measure of literal adherence to the ground truth vocabulary, whereas Judge Accuracy serves as a more robust indicator of functional correctness

5.4 Findings

Our findings suggest that the resource perspective and the control-flow perspective are handled relatively independently by LLMs. This is supported by the results of the paired t-tests conducted to assess the impact of resource discovery on control-flow quality. For all of the evaluated processes, no statistically significant difference in control-flow quality was observed between the baseline and the resource-aware models ($p > 0.05$). This stability indicates that the additional effort of identifying pools and lanes does not interfere with the model's ability to infer the correct sequence of activities, supporting the feasibility of multi-perspective process generation. Additionally, our efficiency analysis demonstrates that this semantic enrichment comes with negligible computational overhead. The runtime distribution reveals only a marginal increase in generation time for some models.

Regarding the resource perspective, the consistently high Judge-based accuracy scores confirm that LLMs possess robust semantic reasoning capabilities for identifying process participants. While the models successfully extract relevant actors, the lower semantic similarity scores in certain cases indicate lexical divergence from the ground truth. These deviations often represent valid alternative modeling decisions, yet they may also be attributed to inherent biases present in the models, similarly to what [Kourani *et al.*, 2025a] observe.

Nevertheless, semantic ambiguity is inherent in the textual descriptions, reflecting the subjective nature of process modeling: a single text can admit multiple valid structures. We address this by using a Judge-LLM accuracy metric, since deterministic measures cannot reliably credit alternative yet semantically defensible modeling choices. Finally, although generation is non-deterministic, prior benchmarking on this dataset [Kourani *et al.*, 2025b] shows that model families exhibit stable average performance, supporting the reliability of the reported trends.

6 Conclusion

This paper extends generative process modeling beyond the control-flow perspective by integrating resource information. Building on ProMoAI, we augment its intermediate representation with explicit pool/lane annotations and introduce a transformation pipeline (POWL $\rightarrow$ BPMN) that produces syntactically correct collaboration diagrams with automated layout.

We evaluated the approach on a curated dataset of ten process descriptions across nine LLMs. Results show that adding the organizational perspective does not harm process logic: control-flow quality remains comparable to the resource-agnostic baseline under paired significance testing. Resource assignments are largely semantically valid under an independent judge, even when models diverge from the ground truth due to alternative naming or abstraction choices. Efficiency remains practical, with only marginal overhead in runtime and self-correction iterations. Overall, the findings support a single-stage generation strategy that assigns semantic interpretation to LLMs while offloading transformation and visualization to dedicated algorithms, enabling future extensions to additional perspectives such as data objects and temporal constraints.

Ethical Statement

There are no ethical issues.

References

[Abbad-Andaloussi *et al.*, 2025] Amine Abbad-Andaloussi, Marco Franceschetti, Hugo A. López, Clemens Schreiber, and Barbara Weber. When a model gives you mixed signals: cognitive effects and visual behavior. *Process Science*, 2(1):19, Oct 2025.

[Berti *et al.*, 2023] Alessandro Berti, Sebastiaan J. van Zelst, and Daniel Schuster. Pm4py: A process mining library for python. *Softw. Impacts*, 17:100556, 2023.

[Bistarelli *et al.*, 2025] Stefano Bistarelli, Marco Fiore, Ivan Mercanti, and Marina Mongiello. Usage of Large Language Model for Code Generation Tasks: A Review. *SN Comput. Sci.*, 6(6):673, 2025.

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[Casella and Berger, 2024] George Casella and Roger Berger. *Statistical inference*. Chapman and Hall/CRC, 2024.

[Corradini *et al.*, 2021] Flavio Corradini, Fabrizio Fornari, Andrea Polini, Barbara Re, and Francesco Tiezzi. Repository: a repository platform for sharing business process models and logs. In *ITBPM@BPM*, volume 2952 of *CEUR Workshop Proceedings*, pages 13–18. CEUR-WS.org, 2021.

[Dumas *et al.*, 2018] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management, Second Edition*. Springer, 2018.

[Jha *et al.*, 2023] Susmit Jha, Sumit Kumar Jha, Patrick Lincoln, Nathaniel D. Bastian, Alvaro Velasquez, and Sandeep Neema. Dehallucinating Large Language Models Using Formal Methods Guided Iterative Prompting. In *ICAA*, pages 149–152. IEEE, 2023.

[Ji *et al.*, 2023] Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. Towards mitigating hallucination in large language models via self-reflection. *CoRR*, abs/2310.06271, 2023.

[Jiang *et al.*, 2024] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A Survey on Large Language Models for Code Generation. *CoRR*, abs/2406.00515, 2024.

[Kitzmann *et al.*, 2009] Ingo Kitzmann, Christoph König, Daniel Lübke, and Leif Singer. A simple algorithm for automatic layout of BPMN processes. In Birgit Hofreiter and Hannes Werthner, editors, *2009 IEEE Conference on Commerce and Enterprise Computing, CEC 2009, Vienna, Austria, July 20-23, 2009*, pages 391–398. IEEE Computer Society, 2009.

[Klievtsova *et al.*, 2025] Nataliia Klievtsova, Matthias Ehrendorfer, Juergen Mangler, and Stefanie Rinderle-Ma. AutoBPMN.AI: Conversational Process Modeling and Automation. *CEUR Workshop Proceedings*, 4032:304–311, 2025.

[Köpke and Safan, 2024] Julius Köpke and Aya Safan. Introducing the BPMN-Chatbot for Efficient LLM-Based Process Modeling. In *BPM (Demos / Resources Forum)*, volume 3758 of *CEUR Workshop Proceedings*, pages 86–90. CEUR-WS.org, 2024.

[Kourani and van Zelst, 2023] Humam Kourani and Sebastiaan J. van Zelst. POWL: Partially Ordered Workflow Language. In *BPM*, volume 14159 of *Lecture Notes in Computer Science*, pages 92–108. Springer, 2023.

[Kourani *et al.*, 2024a] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. Process Modeling with Large Language Models. In *BPMDS/EMMSAD@CAiSE*, volume 511 of *Lecture Notes in Business Information Processing*, pages 229–244. Springer, 2024.

[Kourani *et al.*, 2024b] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. ProMoAI: Process Modeling with Generative AI. In *IJCAI*, pages 8708–8712. ijcai.org, 2024.

[Kourani *et al.*, 2025a] Humam Kourani, Anton Antonov, Alessandro Berti, and Wil M. P. van der Aalst. Knowledge-Driven Hallucination in Large Language Models: An Empirical Study on Process Modeling. *CoRR*, abs/2509.15336, 2025.

[Kourani *et al.*, 2025b] Humam Kourani, Alessandro Berti, Daniel Schuster, and Wil M. P. van der Aalst. Evaluating Large Language Models on Business Process Modeling: Framework, Benchmark, and Self-Improvement Analysis. *Software and Systems Modeling*, 2025.

[Licardo *et al.*, 2025] Josip Tomo Licardo, Nikola Tankovic, and Darko Etinger. BPMN Assistant: An LLM-Based Approach to Business Process Modeling. *CoRR*, abs/2509.24592, 2025.

[Lin *et al.*, 2024] Leilei Lin, Yumeng Jin, Yingming Zhou, Wenlong Chen, and Chen Qian. MAO: A Framework for Process Model Generation with Multi-Agent Orchestration. *CoRR*, abs/2408.01916, 2024.

[Lübke *et al.*, 2021] Daniel Lübke, Maike Ahrens, and Kurt Schneider. Influence of diagram layout and scrolling on understandability of BPMN processes: an eye tracking experiment with BPMN diagrams. *Inf. Technol. Manag.*, 22(2):99–131, 2021.

[Miyake *et al.*, 2025] Daiki Miyake, Akihiro Iohara, Yu Saito, and Toshiyuki Tanaka. Negative-prompt inversion: Fast image inversion for editing with text-guided diffusion models. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2025, Tucson, AZ, USA, February 26 - March 6, 2025*, pages 2063–2072. IEEE, 2025.

[Nivon and Salaün, 2024] Quentin Nivon and Gwen Salaün. Automated Generation of BPMN Processes from Textual Requirements. In *ICSOC (1)*, volume 15404 of *Lecture Notes in Computer Science*, pages 185–201. Springer, 2024.

[OMG, 2011] OMG. Business Process Model and Notation (BPMN), Version 2.0. Technical report, Object Management Group, January 2011.

[Recker, 2010] Jan Recker. Opportunities and constraints: the current struggle with BPMN. *Bus. Process. Manag. J.*, 16(1):181–201, 2010.

[Reimers and Gurevych, 2021] Nils Reimers and Iryna Gurevych. all-minilm-l6-v2 sentence transformer. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2, 2021. HuggingFace model, accessed December 14, 2025.

[Rosemann, 2006] Michael Rosemann. Potential pitfalls of process modeling: part B. *Bus. Process. Manag. J.*, 12(3):377–384, 2006.

[Safan and Köpke, 2025] Aya Safan and Julius Köpke. BPMN-Chatbot++: LLM-Based Modeling of Collaboration Diagrams with Data. In *Proceedings of the BPM*, 2025.

[Sander, 1995] Georg Sander. A Fast Heuristic for Hierarchical Manhattan Layout. In *GD*, volume 1027 of *Lecture Notes in Computer Science*, pages 447–458. Springer, 1995.

[Sola *et al.*, 2022] Diana Sola, Christian Warmuth, Bernhard Schäfer, Peyman Badakhshan, Jana-Rebecca Rehse, and Timotheus Kampik. SAP signavio academic models: A large process model dataset. In *ICPM Workshops*, volume 468 of *Lecture Notes in Business Information Processing*, pages 453–465. Springer, 2022.

[Wang *et al.*, 2023] Bailin Wang, Zi Wang, Xuezhi Wang, Yuan Cao, Rif A. Saurous, and Yoon Kim. Grammar Prompting for Domain-Specific Language Generation with Large Language Models. In *NeurIPS*, 2023.

[Wang *et al.*, 2024] Zhijie Wang, Zijie Zhou, Da Song, Yuheng Huang, Shengmai Chen, Lei Ma, and Tianyi Zhang. Where Do Large Language Models Fail When Generating Code? *CoRR*, abs/2406.08731, 2024.

[Xu *et al.*, 2023] Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. ExpertPrompting: Instructing Large Language Models to be Distinguished Experts. *CoRR*, abs/2305.14688, 2023.

[Zan *et al.*, 2022] Daoguang Zan, Bei Chen, Dejian Yang, Zeqi Lin, Minsu Kim, Bei Guan, Yongji Wang, Weizhu Chen, and Jian-Guang Lou. CERT: Continual Pre-training on Sketches for Library-oriented Code Generation. In *IJCAI*, pages 2369–2375. ijcai.org, 2022.

[Zhou *et al.*, 2023] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *ICLR*. OpenReview.net, 2023.