

Text2BPMN

BPMN File Generator

Yasmin Awad

ValueBlue Task - hiring step n3

Task Instructions

- Build a small command-line tool that converts a natural-language process description into a valid BPMN 2.0 diagram (XML .bpmn file) using a Large Language Model (LLM).

Input:

- A plain-text or Markdown file describing a business process with or without (actors, steps, decisions, outcomes).

Output

- A .bpmn XML file representing the process as a BPMN 2.0 Collaboration/Process.
- Optional: a simple validation report (stdout) stating whether required elements were found (e.g., at least one StartEvent/EndEvent, at least one Pool/Process).

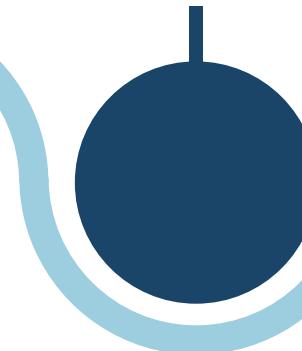
Tech Stack

Langchain
LLM Orchestration

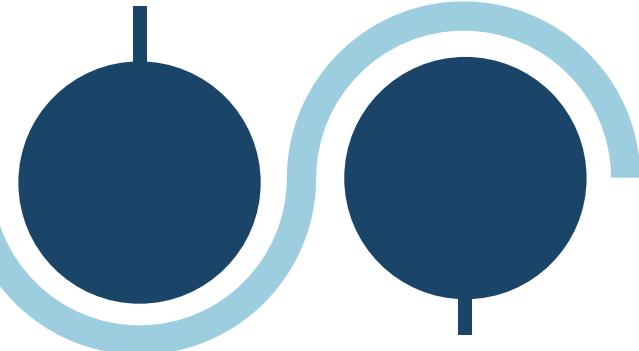


click
CLI framework

AzureOpenAI
LLM Provider (GPT-4.1)



**bpmn
auto-layout**
Diagram layout generation



pytest
Testing framework

User Interface

- Input:**
- Process description as string, .txt, .md
 - Optional: output path for the generated file
- Output:**
- .bpmn file (./output/process_diagram.bpmn or specified location)
 - Reasoning as stdout

```
● (.venv) yasmin@yasmin-Ubuntu25:~/Desktop/YasminAwad_ValueBlueTask/Text2BPMN$ text2bpmn --file ./assets/examples/task2.txt --output ./output/task2.bpmn
=====
💡 Text2BPMN
=====

⚙️ Generating BPMN diagram...
✓ BPMN diagram saved to: ./output/task2.bpmn
→ You can drag and drop the .bpmn file at bpmn.io for visualization

📋 Reasoning Report:
The required elements were not explicitly stated in the process description. A start event ('Receive support request') and end event ('Close support ticket') were created to define the boundaries. The single pool represents the Support Team. Three lanes for escalation levels and one for closure were created to represent different support roles. Each lane contains at least one element. Exclusive gateways were used to model the escalation decisions. All elements have unique IDs, and sequence flows connect all process steps as per BPMN rules.
```

Assumptions

- The input is an English business process description
- Length: 10-10,000 characters
- Supported formats: .txt, .md, or raw text
- Process contains one pool

BPMN File Analysis

```
<?xml version="1.0" encoding="UTF-8"?><bpmn:definitions
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:bpmn="http://www.omg.org/spec/BPMN/20100524/MODEL"
  xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI"
  xmlns:dc="http://www.omg.org/spec/DD/20100524/DC"
  xmlns:di="http://www.omg.org/spec/DD/20100524/DI"
  id="Definitions_1r6uvtt" targetNamespace="http://bpmn.io/
  schema/bpmn" exporter="bpmn-js (https://demo.bpmn.io)"
  exporterVersion="12.0.0">
  <bpmn:collaboration id="Collaboration_leuq2fm">
    <bpmn:participant id="Participant_luu4c6f" name="Pool /
      Participant" processRef="Process_0zjjsxm"/>
  </bpmn:collaboration>
  > <bpmn:process id="Process_0zjjsxm" name="Pool /
    Participant">...
  </bpmn:process>
  > <bpmndi:BPMNDiagram id="BPMNDiagram_1">...
  </bpmndi:BPMNDiagram>
</bpmn:definitions>
```

BPMN File Analysis

```
<bpmn:process id="Process_0zjjsxm" name="Pool / Participant">
  <bpmn:laneSet id="LaneSet_18deus5">
    <bpmn:lane id="Lane_0k6zdn1" name="Customer Support">
      <bpmn:flowNodeRef>Event_0ix3gle</bpmn:flowNodeRef>
      <bpmn:flowNodeRef>Activity_1ruky8s</bpmn:flowNodeRef>
      <bpmn:flowNodeRef>Gateway_0ysw5iv</bpmn:flowNodeRef>
      <bpmn:flowNodeRef>Activity_1tr7rln</bpmn:flowNodeRef>
      <bpmn:flowNodeRef>Event_15ckywo</bpmn:flowNodeRef>
    </bpmn:lane>
    <bpmn:lane id="Lane_171btsa" name="Technical Support">...
    </bpmn:lane>
    <bpmn:lane id="Lane_0saq4lj" name="Advanced Support">...
    </bpmn:lane>
    <bpmn:lane id="Lane_1rtovro" name="Expert Support">...
    </bpmn:lane>
  </bpmn:laneSet>
  <bpmn:startEvent id="Event_0ix3gle" name="Start Support Process">...
  </bpmn:startEvent>
  <bpmn:userTask id="Activity_1ruky8s" name="Initial Support">...
  </bpmn:userTask>
  <bpmn:exclusiveGateway id="Gateway_0ysw5iv" name="Escalation Decision">...
  </bpmn:exclusiveGateway>
  <bpmn:userTask id="Activity_0ix8fan" name="Level 1 Escalation">...
  </bpmn:userTask>
  <bpmn:userTask id="Activity_0mvbj06" name="Level 2 Escalation">...
  </bpmn:userTask>
  <bpmn:userTask id="Activity_0bgz04y" name="Level 3 Escalation">...
  </bpmn:userTask>
  <bpmn:userTask id="Activity_1tr7rln" name="Resolution">...
  </bpmn:userTask>
  <bpmn:endEvent id="Event_15ckywo" name="End Support Process">...
  </bpmn:endEvent>
  <bpmn:sequenceFlow id="Flow_0szra3d" sourceRef="Event_0ix3gle" targetRef="Activity_1ruky8s"/>
  <bpmn:sequenceFlow id="Flow_11gvh68" sourceRef="Activity_1ruky8s" targetRef="Gateway_0ysw5iv"/>
  <bpmn:sequenceFlow id="Flow_lucxyuh" name="Issue requires Level 1 escalation" sourceRef="Gateway_0ysw5iv" targetRef="Activity_0ix8fan"/>
  <bpmn:sequenceFlow id="Flow_0j6haZ8" name="Issue requires Level 2 escalation" sourceRef="Gateway_0ysw5iv" targetRef="Activity_0mvbj06"/>
  <bpmn:sequenceFlow id="Flow_0ley814" name="Issue requires Level 3 escalation" sourceRef="Gateway_0ysw5iv" targetRef="Activity_0bgz04y"/>
  <bpmn:sequenceFlow id="Flow_0ozpldq" name="Issue resolved without escalation" sourceRef="Gateway_0ysw5iv" targetRef="Activity_1tr7rln"/>
  <bpmn:sequenceFlow id="Flow_0jlj3lu" sourceRef="Activity_0ix8fan" targetRef="Activity_1tr7rln"/>
  <bpmn:sequenceFlow id="Flow_0727a30" sourceRef="Activity_0mvbj06" targetRef="Activity_1tr7rln"/>
  <bpmn:sequenceFlow id="Flow_0py3yk7" sourceRef="Activity_0bgz04y" targetRef="Activity_1tr7rln"/>
  <bpmn:sequenceFlow id="Flow_1j8lavq" sourceRef="Activity_1tr7rln" targetRef="Event_15ckywo"/>
</bpmn:process>
```

BPMN File Analysis

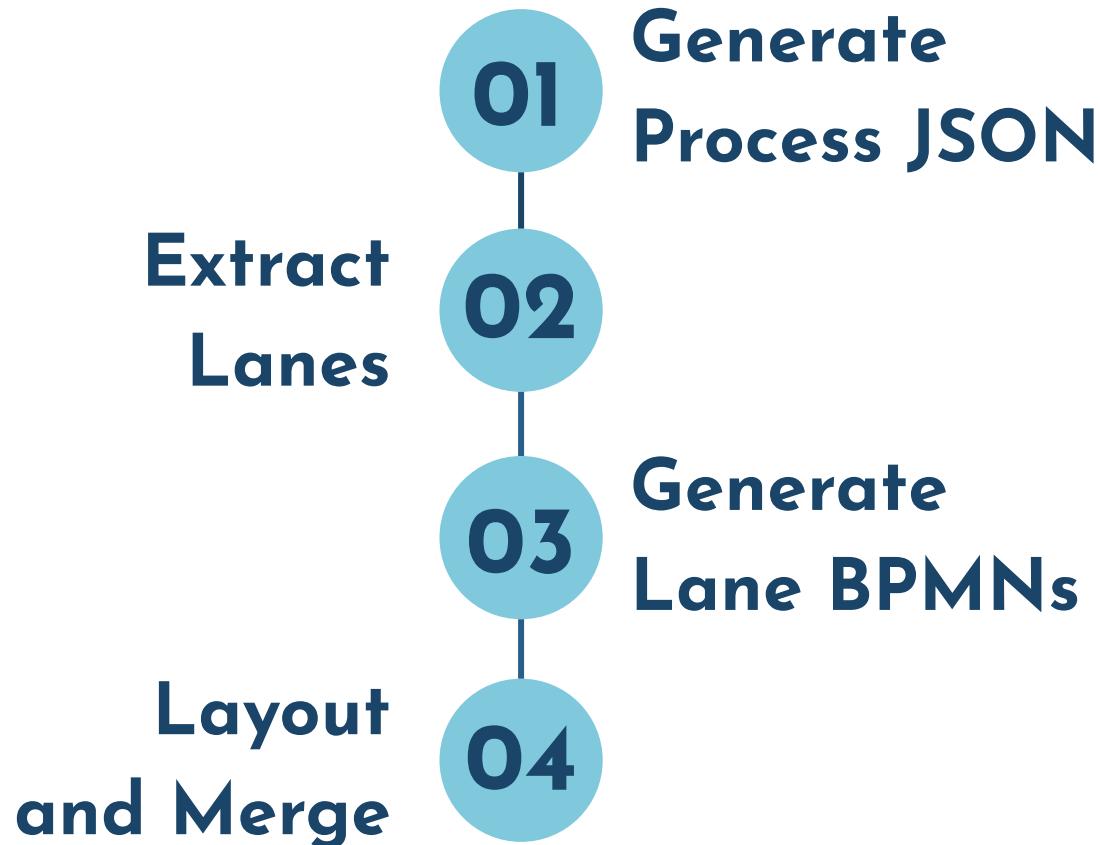
```
<bpmdi:BPMDiagram id="BPMDiagram_1">
  <bpmdi:BPMPNPlane id="BPMPNPlane_1" bpmnElement="Collaboration_leuq2fm">
    <bpmdi:BPMSHape id="Participant_luu4c6f_di" bpmnElement="Participant_luu4c6f" isHorizontal="true">
      <dc:Bounds x="100" y="-25" width="1068" height="1000"/>
      <bpmdi:BPMLabel/>
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Lane_0k6zdni_di" bpmnElement="Lane_0k6zdni" isHorizontal="true">
      <dc:Bounds x="130" y="-25" width="1038" height="250"/>
      <bpmdi:BPMLabel/>
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Lane_171btSA_di" bpmnElement="Lane_171btSA" isHorizontal="true">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Lane_0saq41j_di" bpmnElement="Lane_0saq41j" isHorizontal="true">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Lane_1rtovro_di" bpmnElement="Lane_1rtovro" isHorizontal="true">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Event_0ix3gle_di" bpmnElement="Event_0ix3gle">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Activity_1rukY8s_di" bpmnElement="Activity_1rukY8s">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Gateway_0ysw5iv_di" bpmnElement="Gateway_0ysw5iv" isMarkerVisible="true">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Activity_0ix8fan_di" bpmnElement="Activity_0ix8fan">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Activity_0mvbj06_di" bpmnElement="Activity_0mvbj06">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Activity_0bgz04y_di" bpmnElement="Activity_0bgz04y">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Activity_1tr7rln_di" bpmnElement="Activity_1tr7rln">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMSHape id="Event_15ckywo_di" bpmnElement="Event_15ckywo">...
    </bpmdi:BPMSHape>
    <bpmdi:BPMPNEdge id="Flow_0szra3d_di" bpmnElement="Flow_0szra3d">...
    </bpmdi:BPMPNEdge>
    <bpmdi:BPMPNEdge id="Flow_11gvh68_di" bpmnElement="Flow_11gvh68">...
    </bpmdi:BPMPNEdge>
    <bpmdi:BPMPNEdge id="Flow_1ucxyuh_di" bpmnElement="Flow_1ucxyuh">...
    </bpmdi:BPMPNEdge>
    <bpmdi:BPMPNEdge id="Flow_0j6haz8_di" bpmnElement="Flow_0j6haz8">...
    </bpmdi:BPMPNEdge>
    <bpmdi:BPMPNEdge id="Flow_1ley8i4_di" bpmnElement="Flow_1ley8i4">...
    </bpmdi:BPMPNEdge>
```

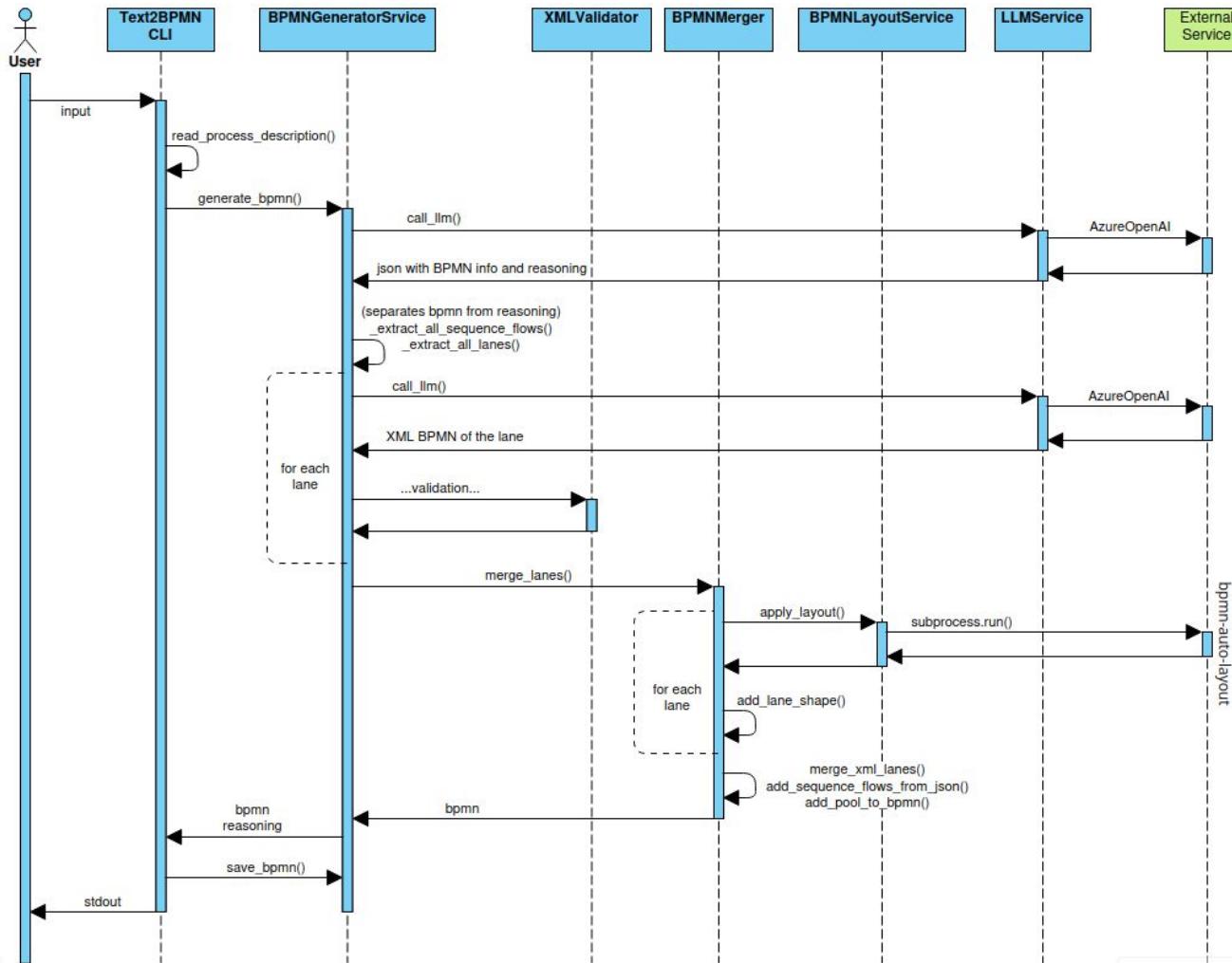
Approach

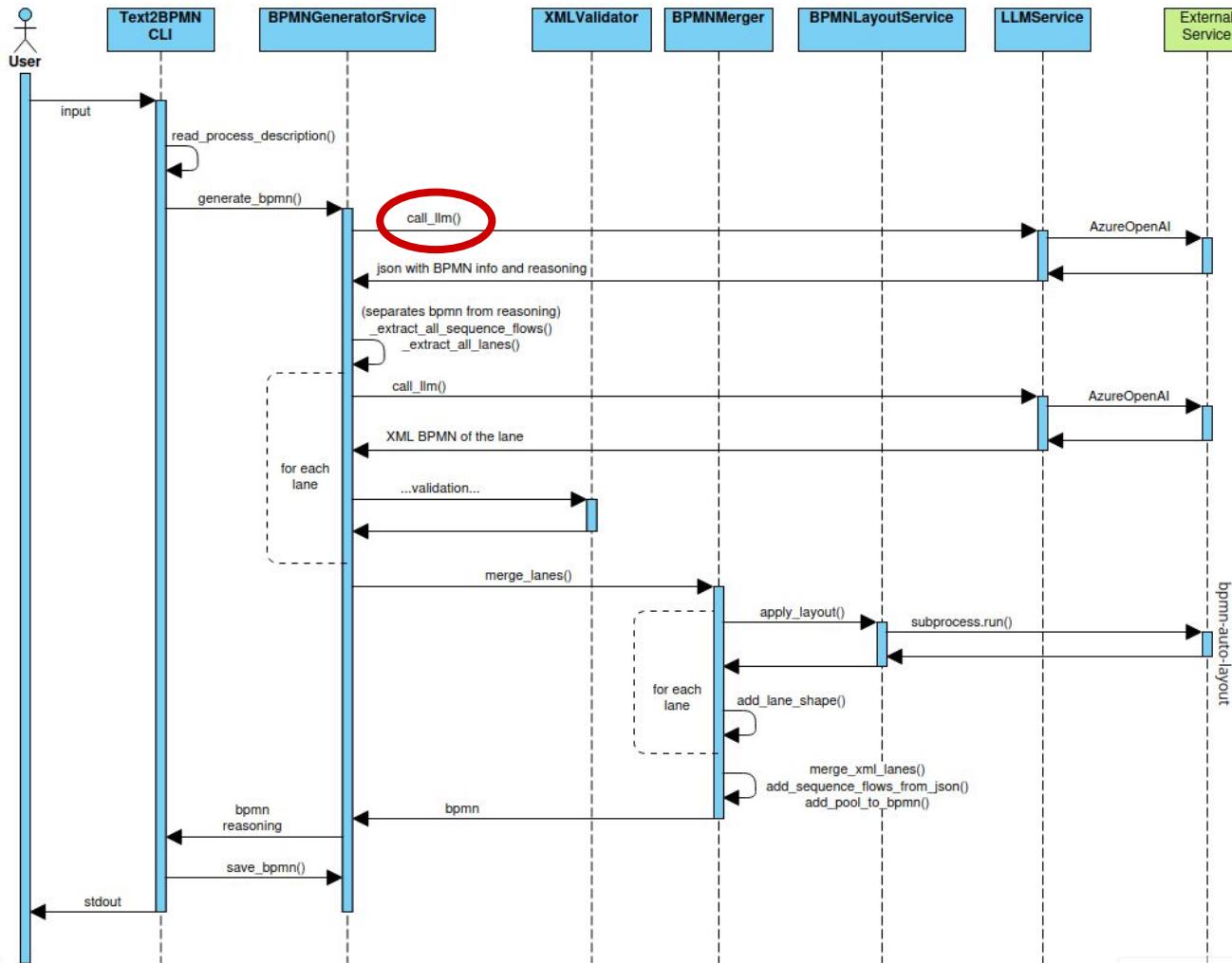
The tool is built around 5 core services:

- **BPMNGeneratorService**: Orchestrates the entire BPMN generation pipeline
- **LLMService**: Manages LLM API configuration and prompt execution
- **BPMNMerger**: Combines individual lane XMLs into a complete BPMN file
- **BPMNLayoutService**: Applies automatic layout (diagram) to single lane bpmn files
- **XMLValidator**: Static utilities for validating XML integrity

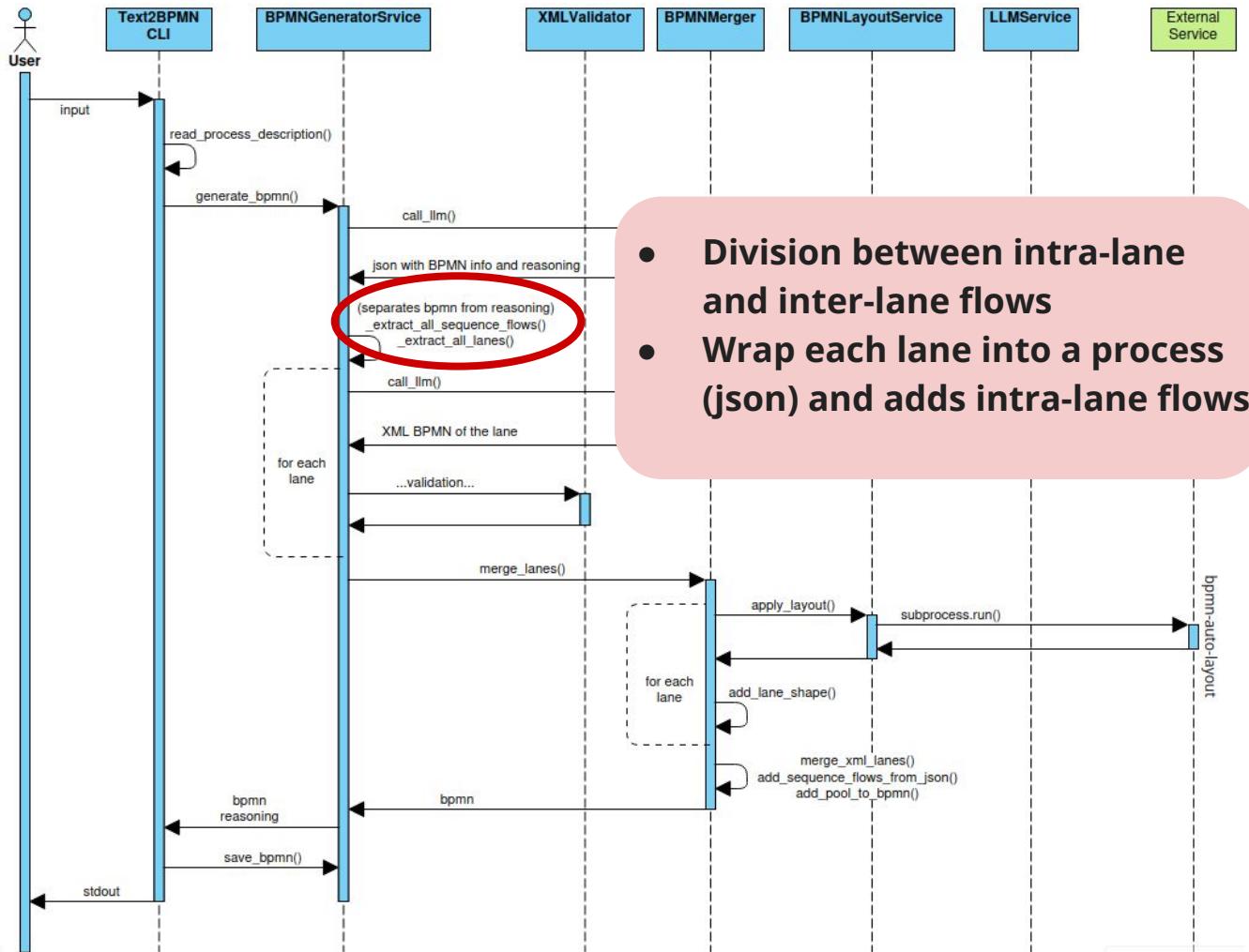
Pipeline Overview

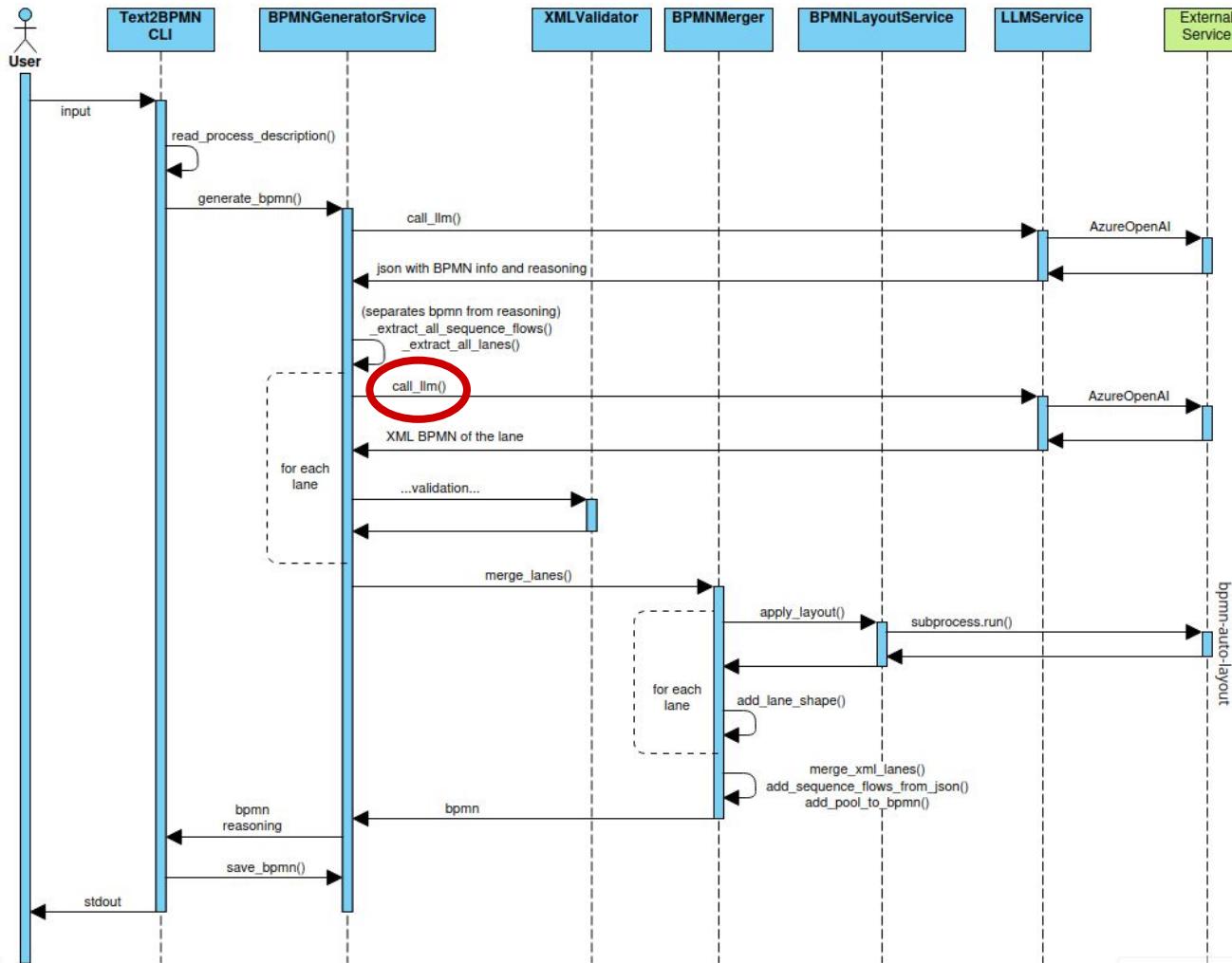


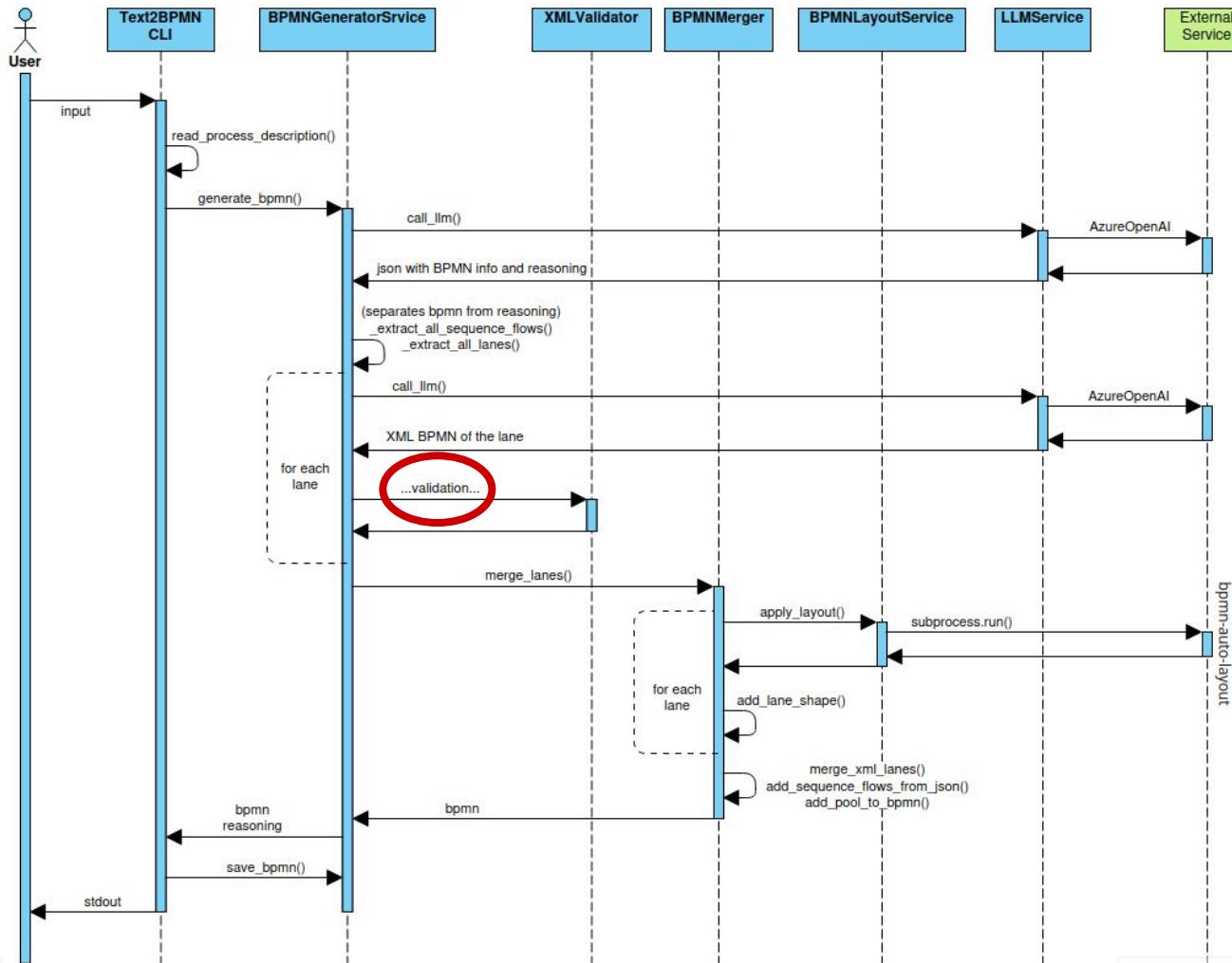


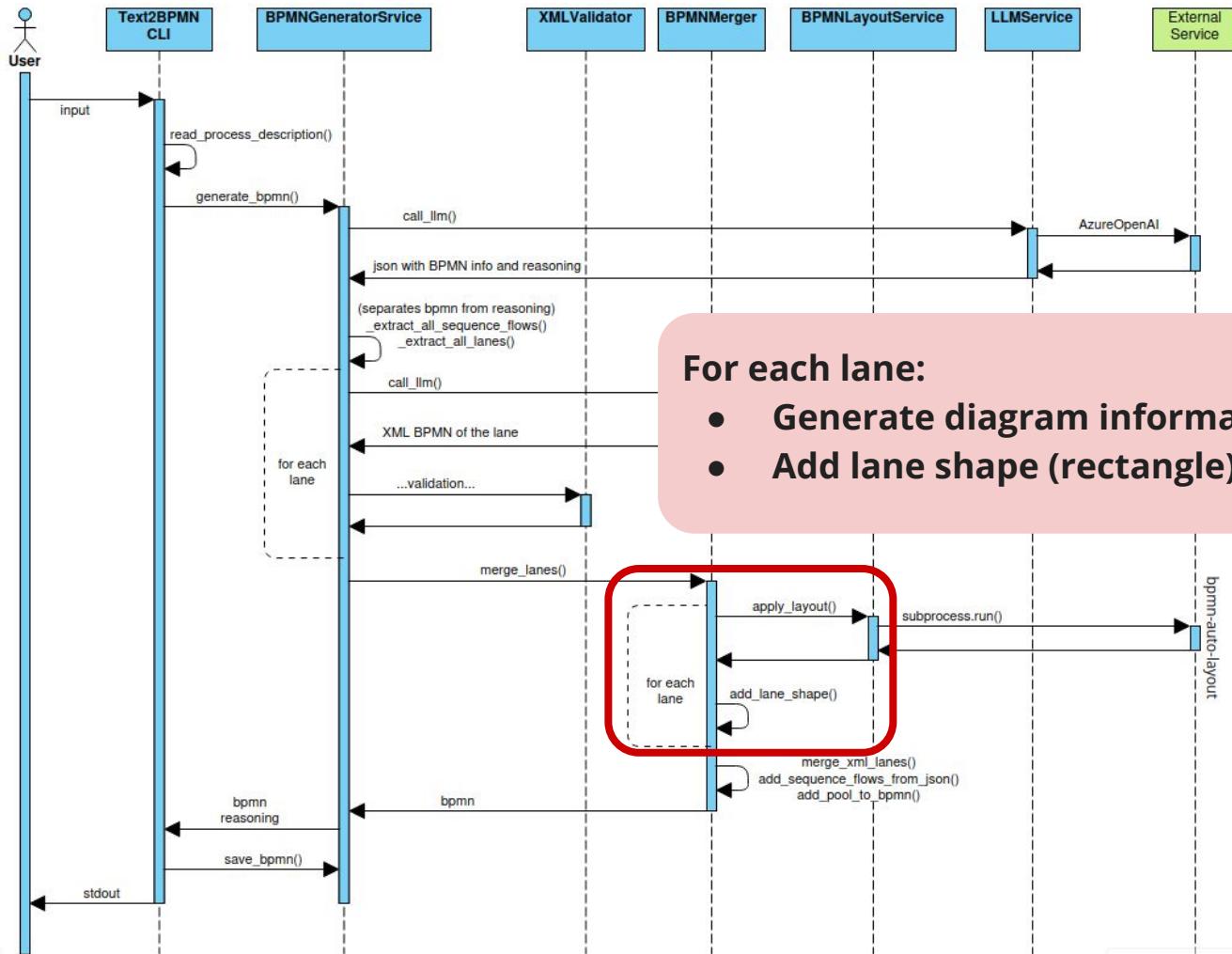


```
{
  "bpmn": {
    "process": {
      "id": "string (use snake_case)",
      "name": "string (human-readable process name)",
      "pool": {
        "id": "string (unique identifier)",
        "name": "string (actor/department name)"
        "lanes": [
          {
            "id": "string (unique identifier)",
            "name": "string (sub-actor or role)",
            "order": 0,
            "elements": [
              {
                "id": "string (unique identifier)",
                "type": "startEvent|endEvent|task|exclusiveGateway|inclusiveGateway|parallelGateway|intermediateEvent",
                "name": "string (short, action-based for tasks)",
                "eventType": "string (for events: none|message|timer|error|conditional)"
              },
            ]
          }
        ],
        "sequenceFlows": [
          {
            "id": "string (unique identifier)",
            "sourceRef": "identifier of the source element",
            "targetRef": "identifier of the target element",
            "name": "string (optional - condition label for gateway branches)",
            "conditionExpression": "string (optional - for XOR branches)"
          }
        ]
      }
    }
  },
  "reasoning": "a validation report (states if the required elements (at least one StartEvent, EndEvent, participant, and process) are stated in the process description. If they are not stated, write how your reasoning created one. If stated, write their correspondent natural language name in the process description)"
}
```



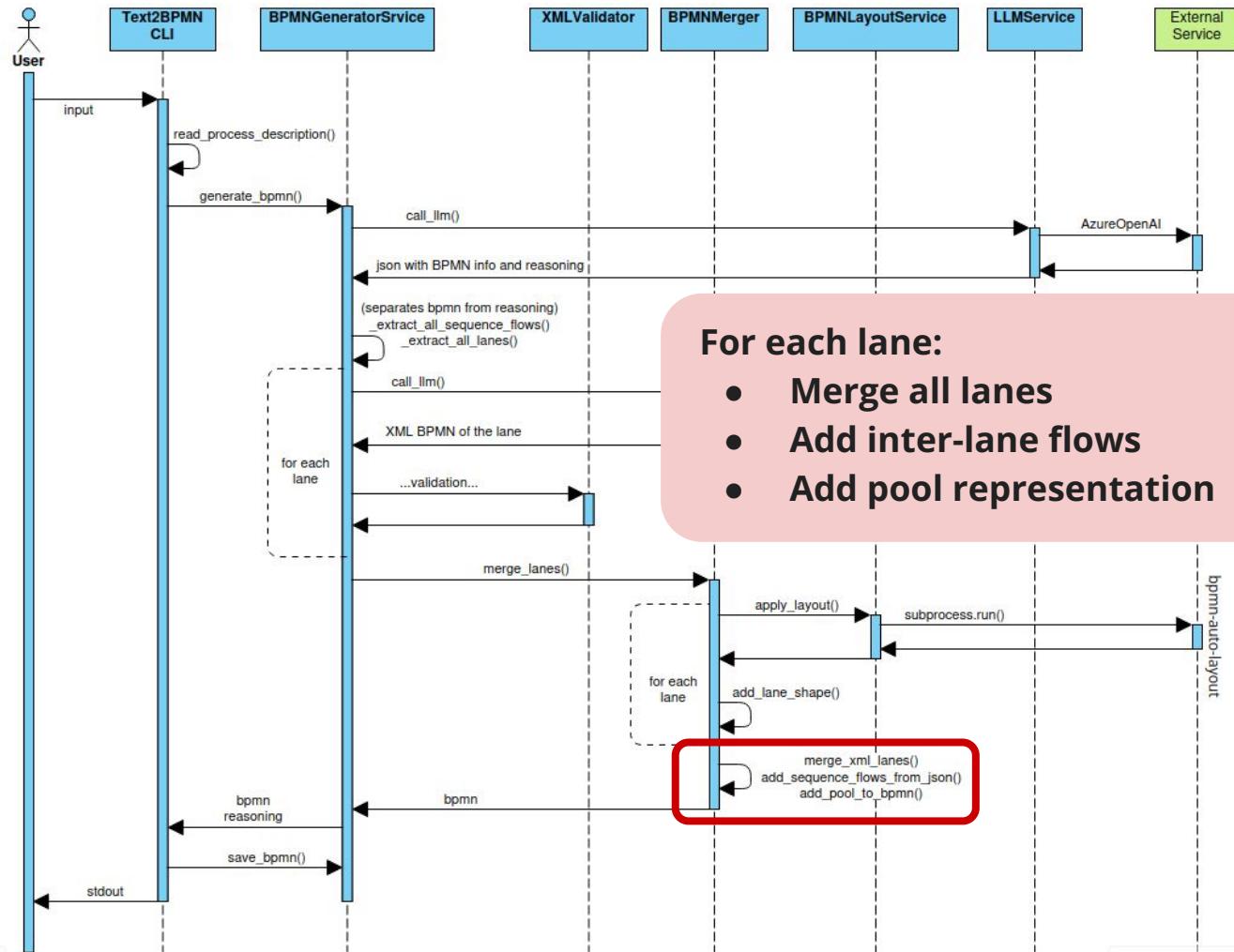






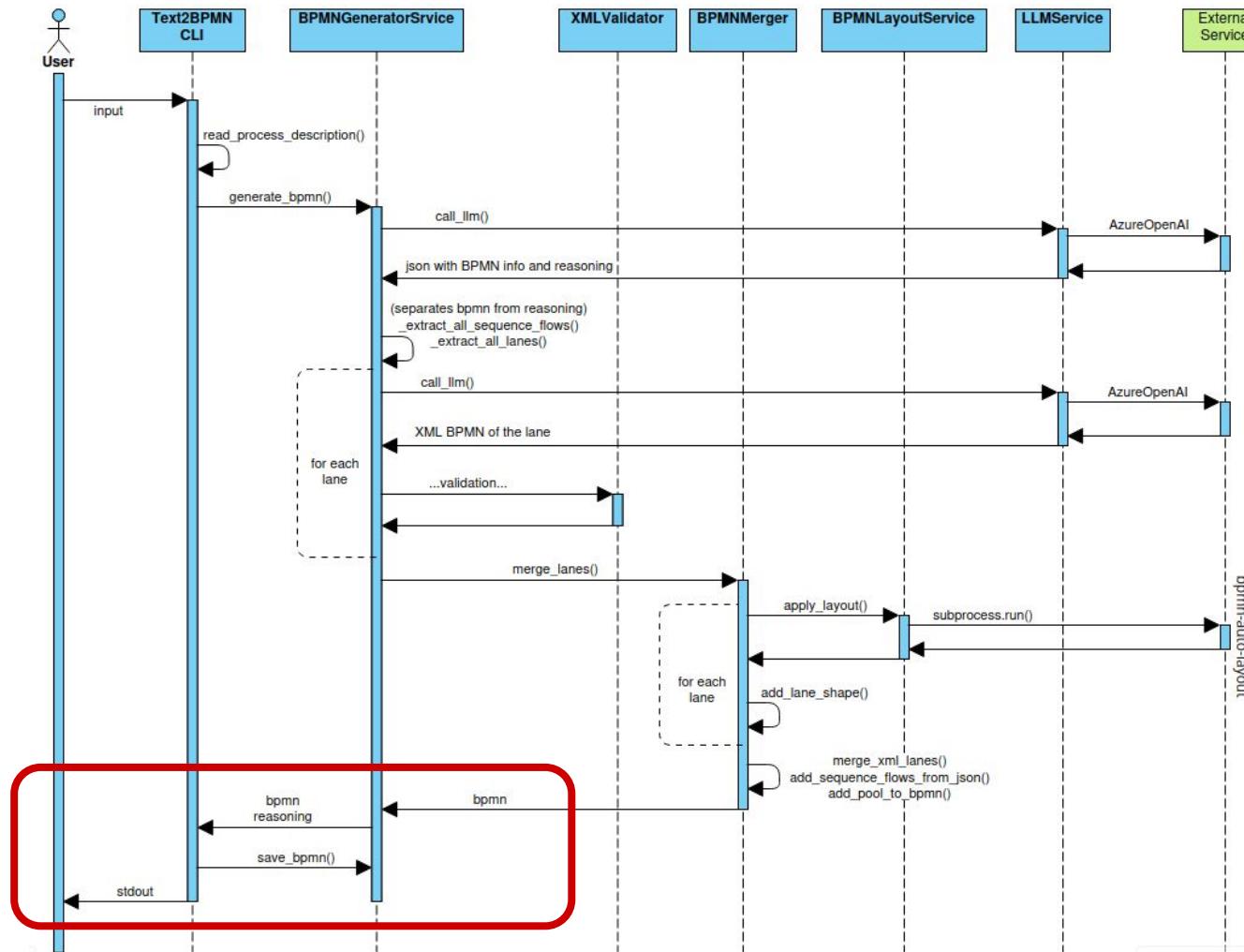
For each lane:

- Generate diagram information
- Add lane shape (rectangle)



For each lane:

- Merge all lanes
 - Add inter-lane flows
 - Add pool representation



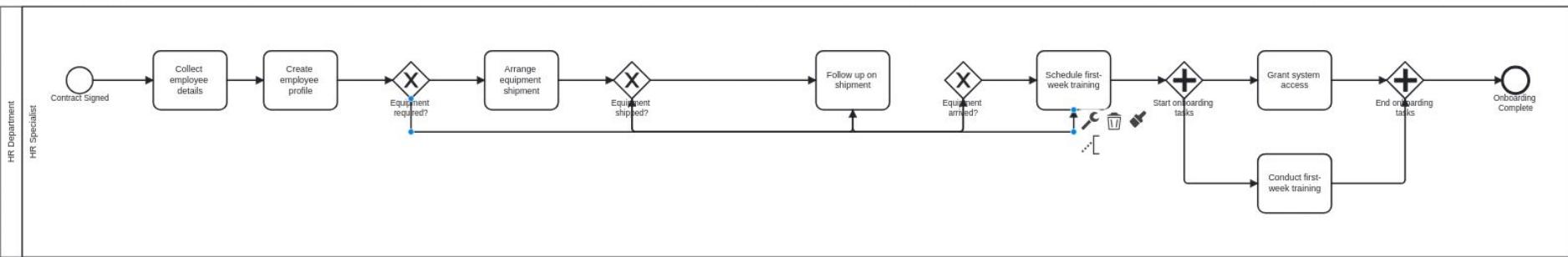
Example 1 - New Employee

Explain the process for onboarding a new remote employee, from contract signing to equipment shipment and first-week training.

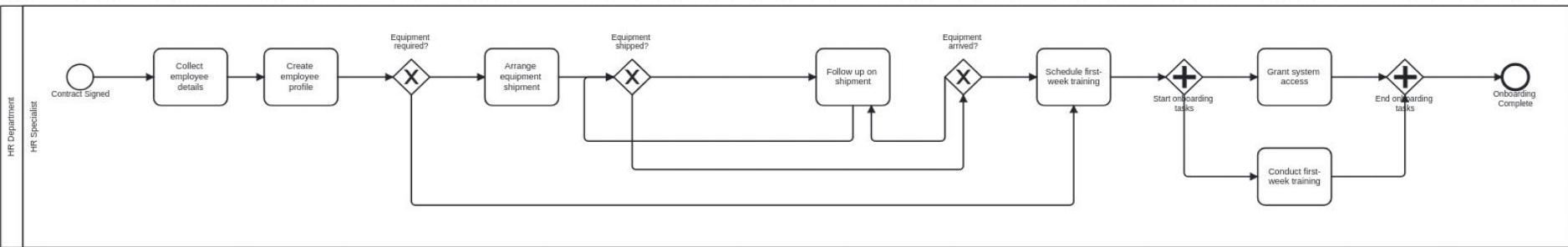
Output - Reasoning

Process description did not explicitly name StartEvent or EndEvent, but the natural process boundary is 'contract signing' (start) and end of first week onboarding (end). The main actor is HR Department; only one lane (HR Specialist) was necessary since all tasks are HR responsibilities. All required BPMN elements are included: one StartEvent, one EndEvent, at least one participant, and a clearly defined process.

Output - BPMN



Output - BPMN



Example 2 - Order Fulfillment

Customer places an online order for products.

The system validates payment information.

If payment is approved, check inventory availability.

If items are in stock, prepare shipment and send to warehouse.

The warehouse packages items and ships to customer.

Customer receives tracking number via email.

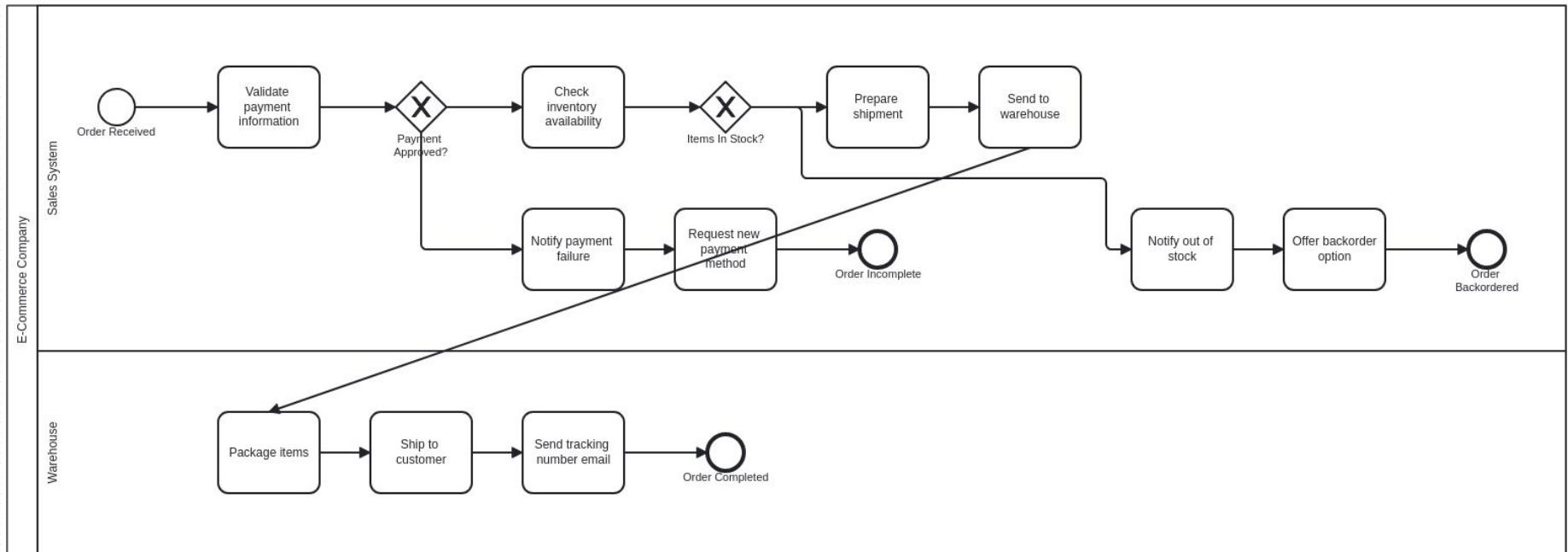
If payment fails, notify customer and request new payment method.

If items are out of stock, notify customer and offer backorder option.

Output - Reasoning

The process description states the triggering event as 'Customer places an online order for products,' which is represented as the Start Event 'Order Received.' Completion events are: successful shipment and notification to the customer (endEvent_order_completed_1), order backordered (endEvent_out_of_stock_1), and payment failure (endEvent_payment_failed_1). The primary actor is the company, with Sales System and Warehouse as internal roles, thus two lanes are used. All tasks, gateways, and flows are derived directly from the description, and all required BPMN elements are present.

Output - BPMN



Experiments

001 - Baseline

Approach:

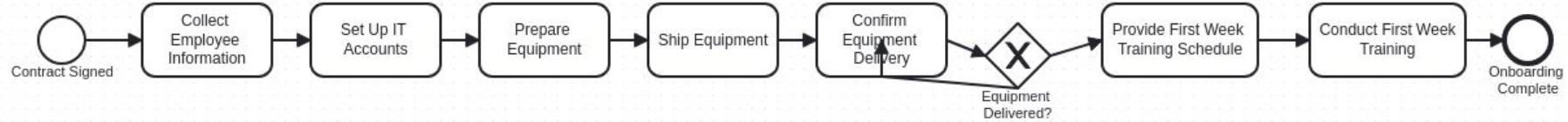
- Prompt LLM directly with BPMN general rules + process description to generate a BPMN XML file
- Added XML validation of the output in order to obtain BPMN

Results:

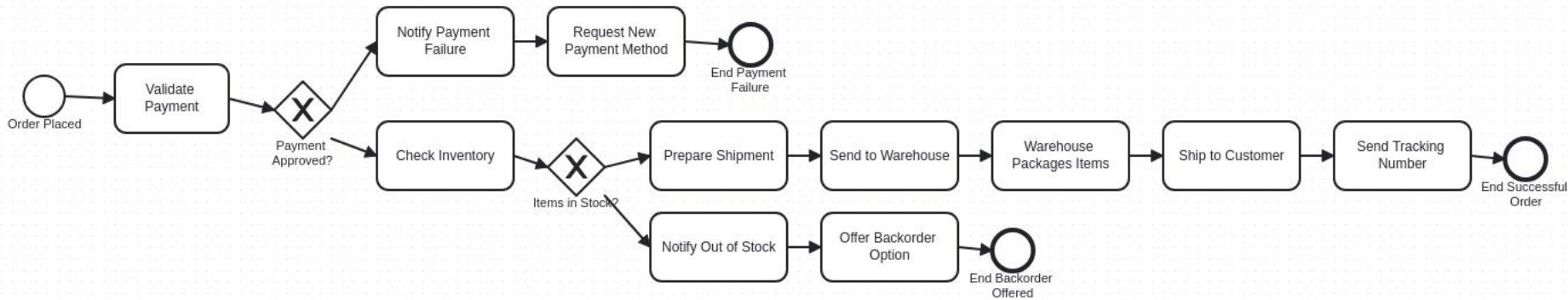
- Model produced plausible events/tasks names in the XML
- Files could be opened in bpmn.io
- Major issues with diagram layout especially if the process is complex and long:
 - Arrows starting/ending in empty space
 - Overlapping elements/arrows
 - Disconnected components

Insight: LLMs understand BPMN concepts but struggle with precise spatial positioning.

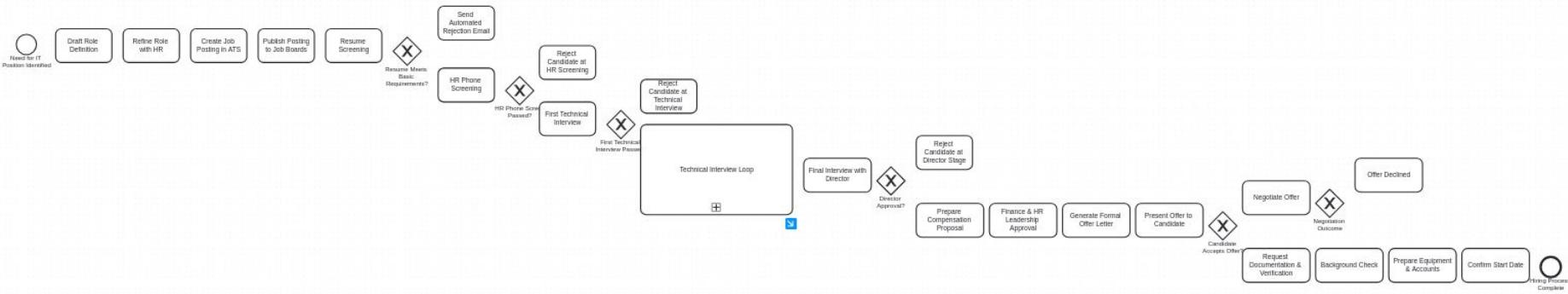
Example 1 - New Employee



Example 2 - Order Fulfillment



Example 3 - Long Hiring Process



002 - From JSON to XML

Approach:

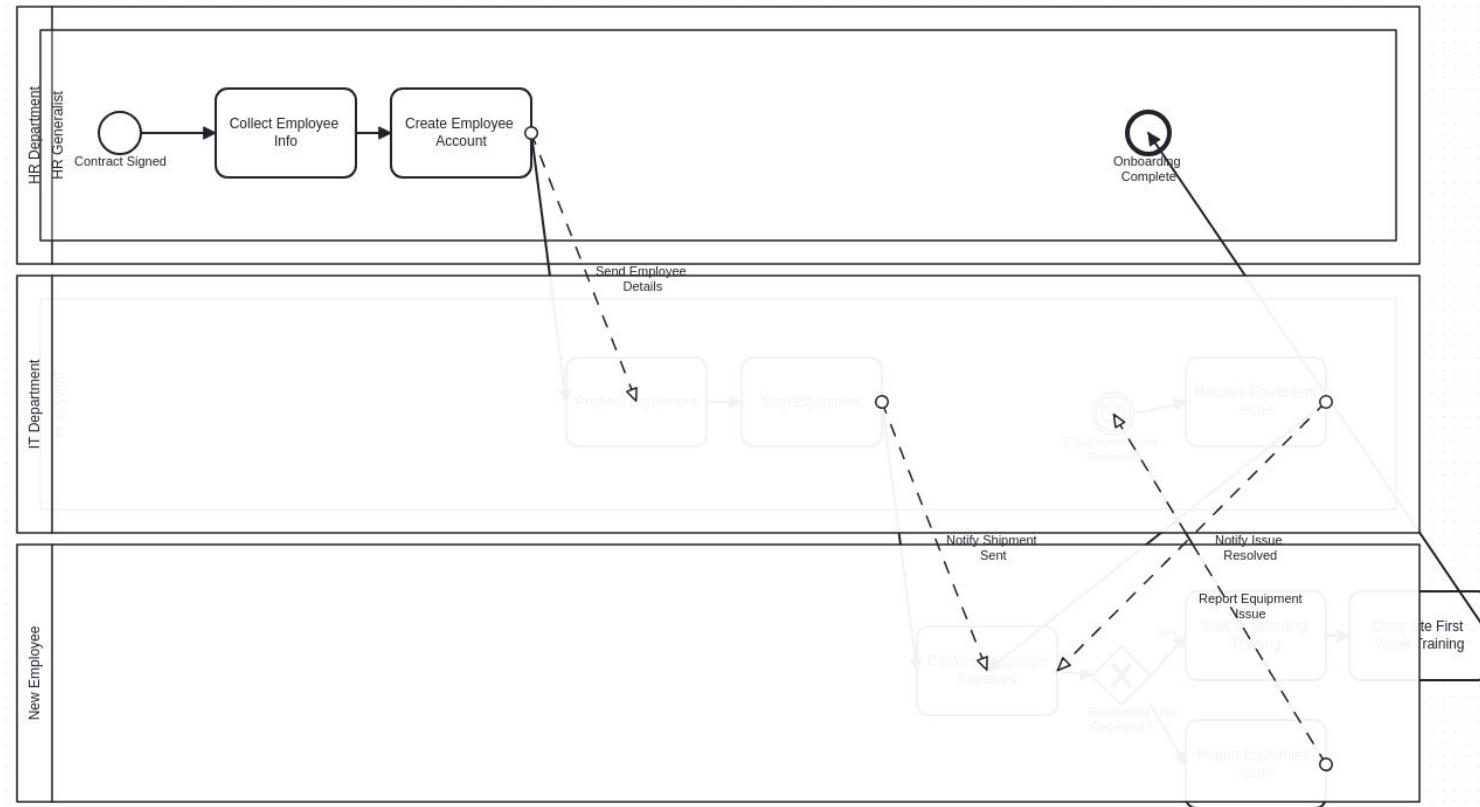
- LLM generates structured JSON
- LLM converts JSON to BPMN XML

Results:

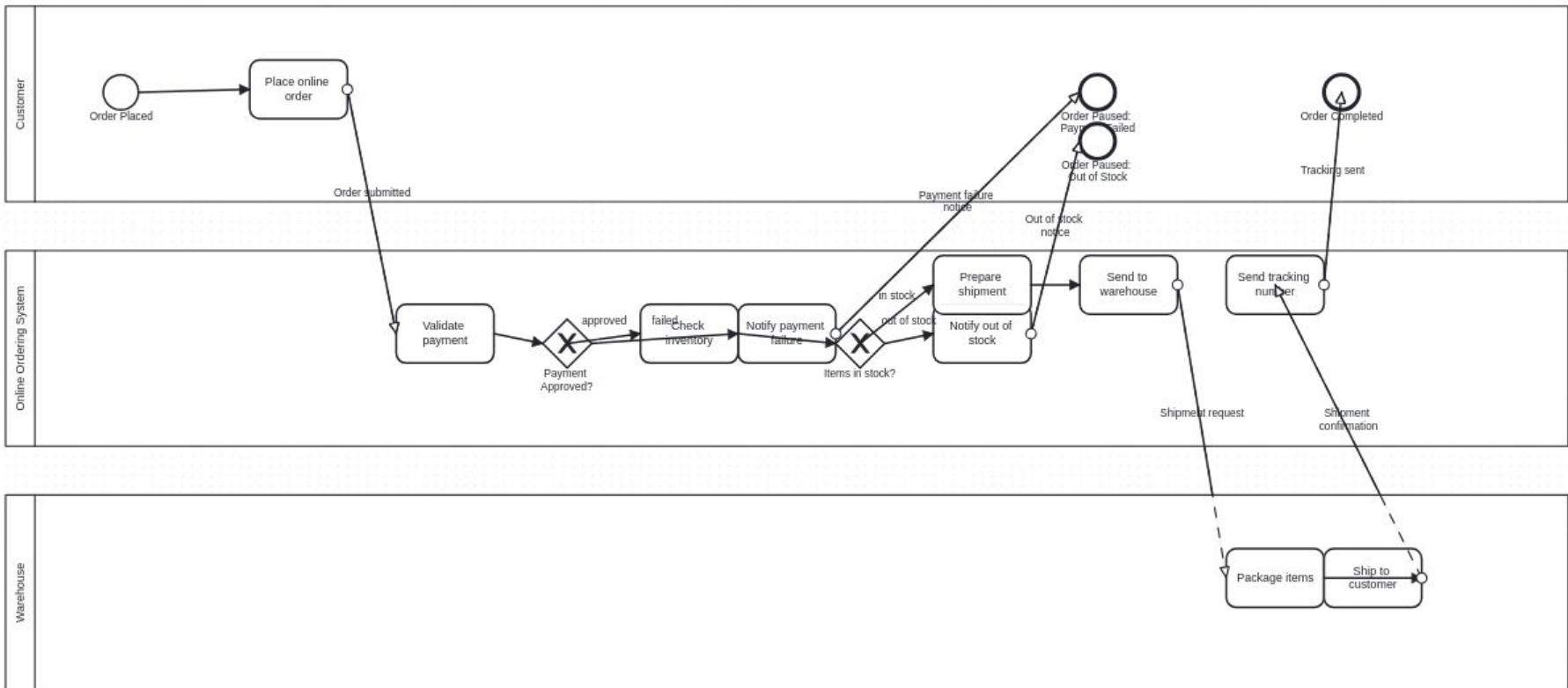
- Initial attempt with full process: Still messy layouts
- Lane-by-lane generation: Some improvement but graphical issues persisted
- Attempted XML of lanes merging via LLM: Failed to produce coherent diagrams

Insight: The graphical/spatial aspect of BPMN is fundamentally difficult for LLMs. A programmatic solution was needed.

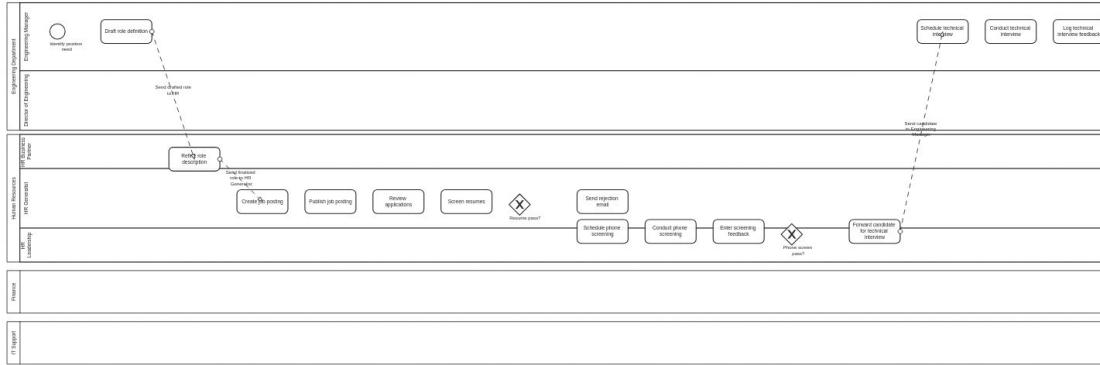
Example 1 - New Employee



Example 2 - Order Fulfillment



Example 3 - Long Hiring Process



003 - bpmn-auto-layout

Approach: Use the [bpmn-auto-layout](#) Node.js library to handle diagram positioning.

Challenge:

The library cannot handle multiple lanes in a single diagram.

Limitations

- Given a collaboration only the first participant's process will be laid out
- Sub-processes will be laid out as collapsed sub-processes
- The following elements are not laid out:
 - Groups
 - Text annotations
 - Associations
 - Message flows

This limitation led to the **divide-and-conquer** approach.

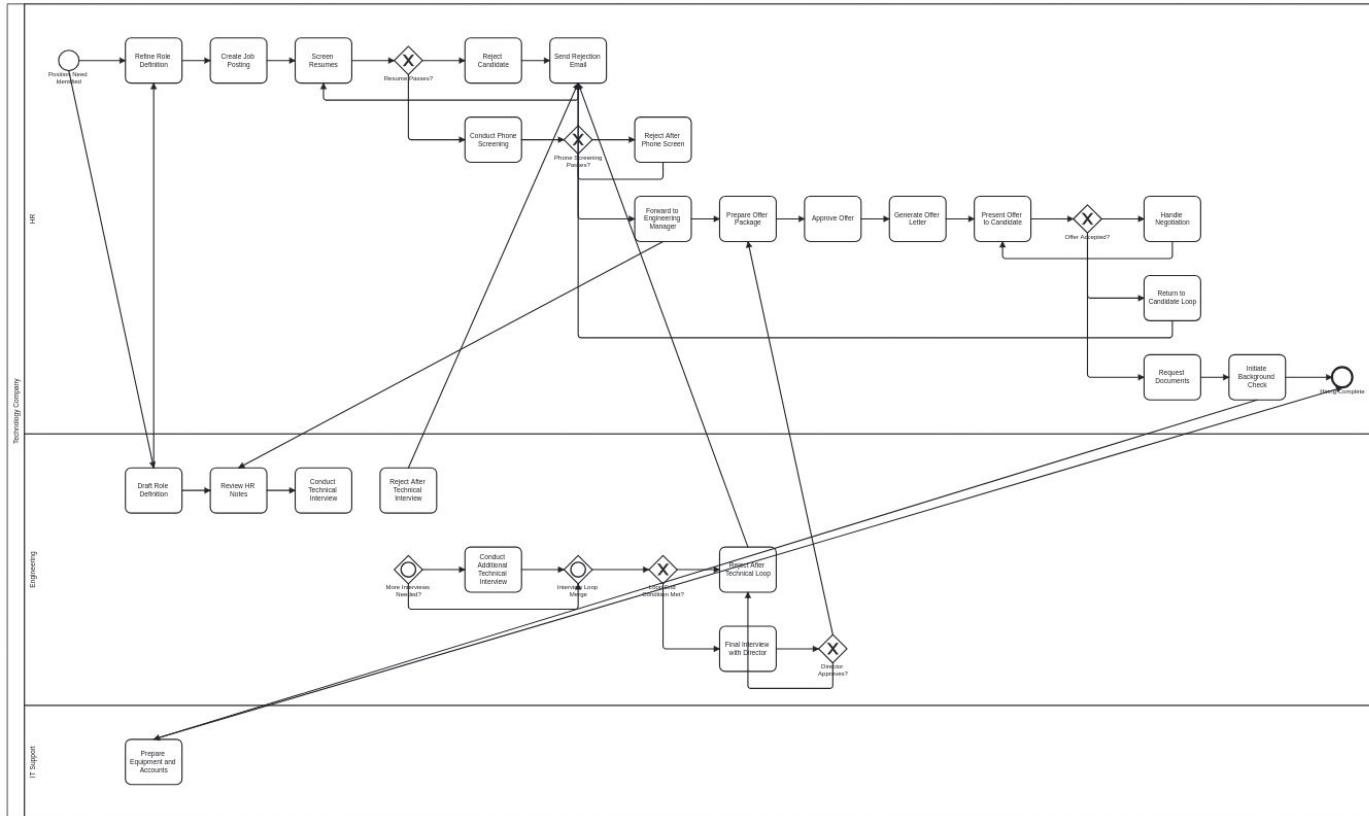
004 - Divide-and-Conquer

Results: Significantly improved visual clarity and structural validity.

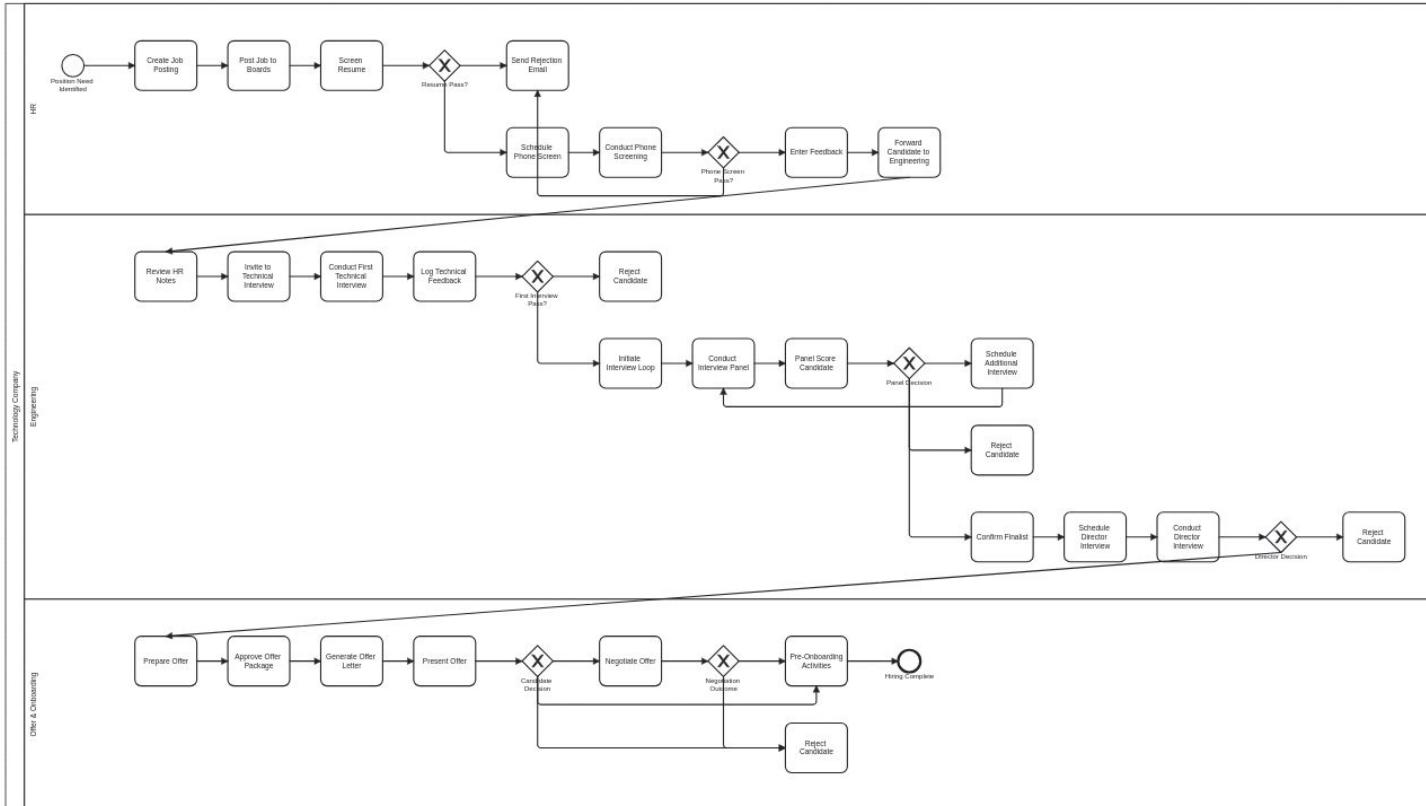
Insight: To ensure a correct LLM generation of xml lanes, it helped a lot instructing the llm in a explicit way to:

- Close definition tags
- Request no collaboration elements in lane XMLs
- Provide targeted examples for each generation stage

Example 3 - Long Hiring Process



Example 3 - NEW Long Hiring Process



004 - Divide-and-Conquer

Limitations:

- **Single-pool constraint** (multi pool processes will be represented by a single pool)
- **Limited BPMN elements:**
 - No extended task types (UserTask, SendTask, etc.)
 - No subprocesses
 - No annotations, associations, or groups
 - No activity calls or transactions
- **Mock start/end events** are removed, but their sequence flows remain

004 - Divide-and-Conquer

Limitations:

- **Arrows overlap:**
 - Intra-lane arrows overlap with each
 - Inter-lane arrows are straight
- **Long Processes Issues:**

A process with a very long and complex description outputs diagrams that are wrong in some steps, or hard to read.

Rejected Approaches

- **Preprocessing step**
Attempted to have the LLM reformulate the process description for clarity. This degraded results rather than improving them.
- **Iterative refinement**
Asking the LLM to review and fix its own XML output did not yield improvements.
- **Parameter tuning**
Changing temperature had minimal impact (fixed LLM configuration for all requests)

Evaluation

I evaluated outputs using three key dimensions:

- **Readability:** Are diagrams clear and non-overlapping?
- **Completeness:** Do all described activities appear?
- **Structural validity:** Start/end events, gateways, sequence flows behave correctly

Evaluation

I evaluated outputs using three key dimensions:

- **Readability:** Are diagrams clear and non-overlapping?
- **Completeness:** Do all described activities appear?
- **Structural validity:** Start/end events, gateways, sequence flows behave correctly

I evaluated in a non-quantitative way:

- Manually reviewed generated diagrams against various test inputs
- Showed results to colleagues unfamiliar with the project
- Since there's no single "correct" BPMN representation, I compared outputs from different pipeline approaches rather than against a ground truth.

Lessons Learned

- **LLMs struggle with spatial positioning**
GPT-4.1 understands process descriptions well but makes errors in exact XML syntax and spatial positioning
- **Layout is a hard problem**
Graphical representation requires geometric reasoning that's poorly suited to text-based LLMs. Specialized tools work better
- **Validation is crucial**
Step-by-step validation prevents cascading errors
- **Examples are powerful**
Providing examples in prompts improved output quality
- **Decomposition works**
Breaking the problem into smaller subproblems

Next Steps

- Clean up mock artifacts
- Strengthen validation rules (deeper BPMN XML rule checking)
- Provide more/better examples to LLM for both JSON and XML generation
- Add multi-pool support
- Expand supported BPMN elements
- Input validation and clarification
- Substitute/Improve the bpmn-auto-layout

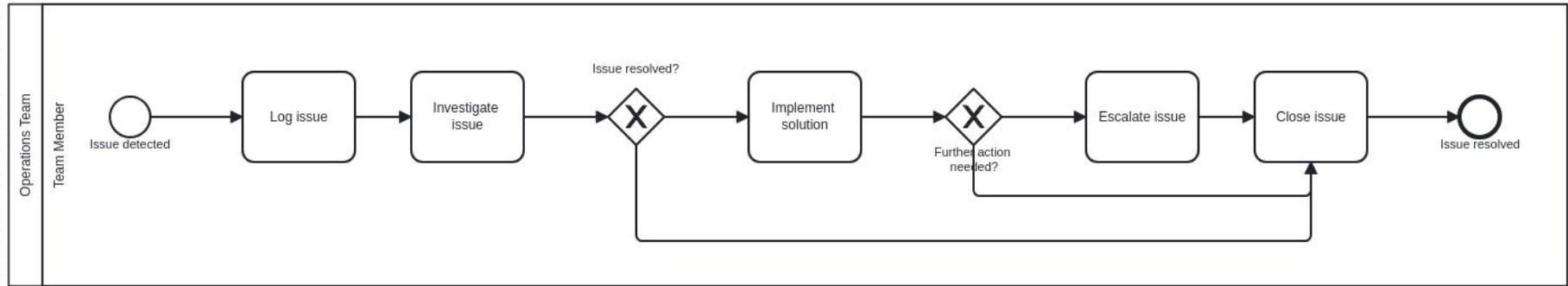
In order to establish better quality metrics we could take into consideration the collaboration with BPMN users.

Thank You

Do you have any questions?

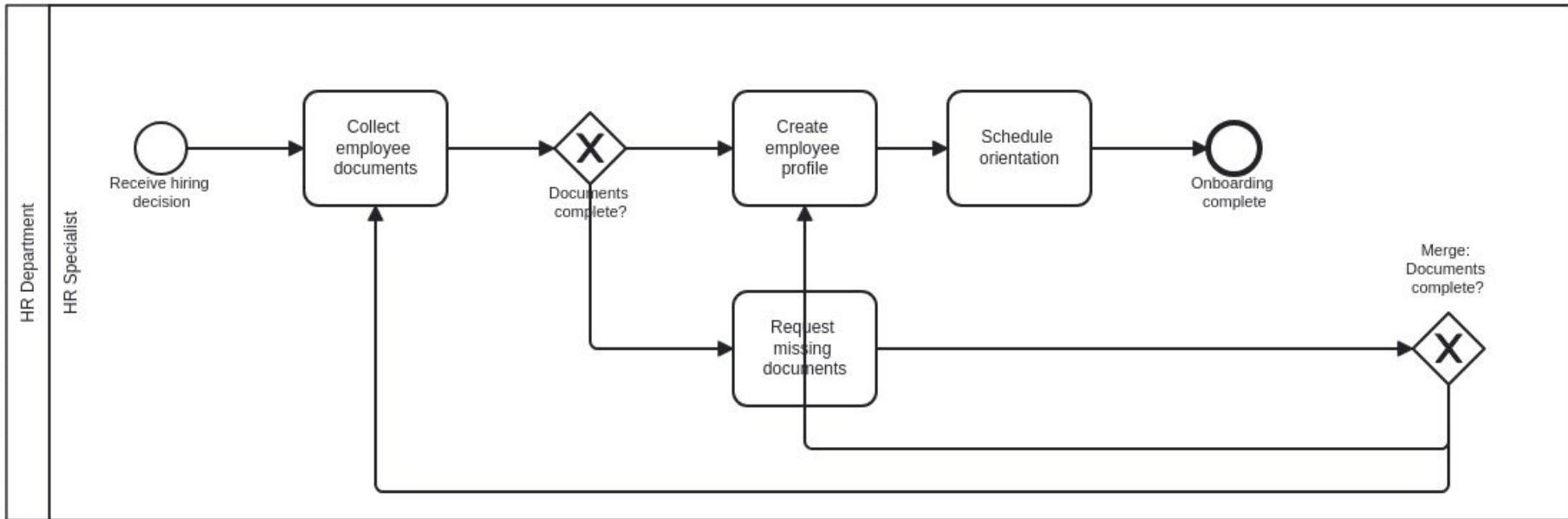
Ambiguous 1

Describe how the team usually deals with issues that sometimes occur during operations.



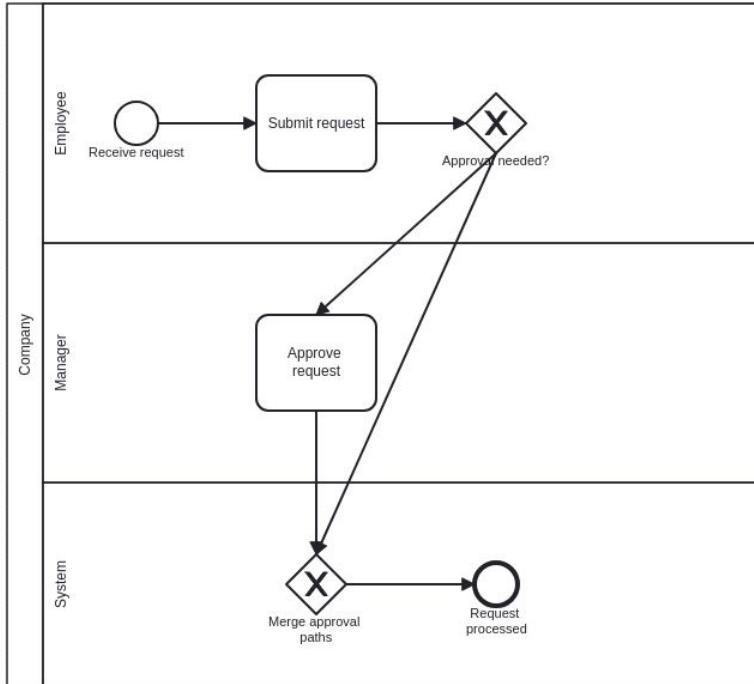
Ambiguous 2

Explain the onboarding flow for new resources according to what we discussed last week.



Contradiction 1

Explain the process where the manager must always approve the request, except when no approval is needed.



Contradiction 2

Describe the workflow where customer complaints must be resolved within 24 hours but cannot be processed the same day.

