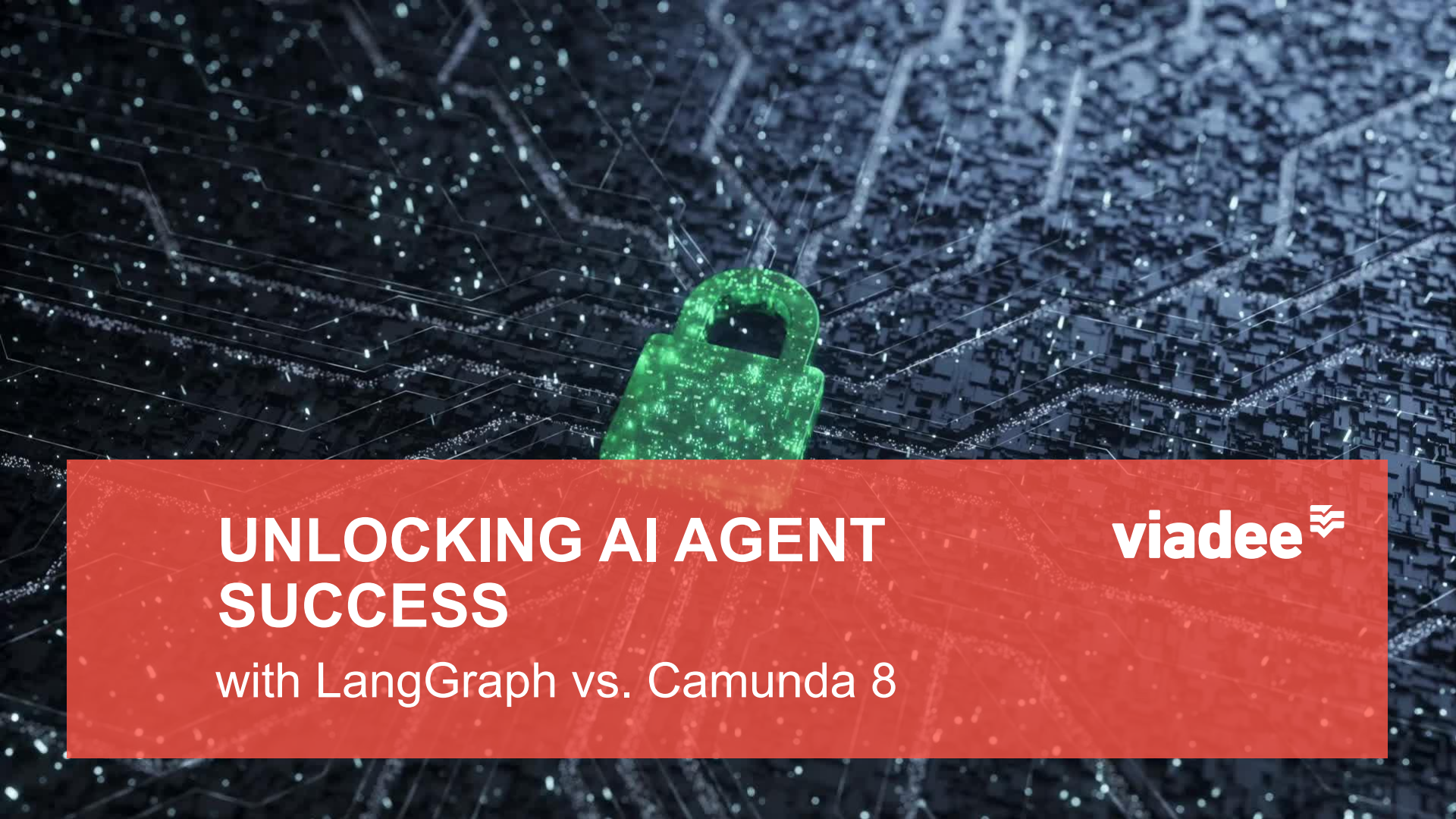


UNLOCKING AI AGENT SUCCESS

with LangGraph vs. Camunda 8

viadee

Who we are

viadee

Mario Micudaj

BPM & AI Consultant

- Data Science, BPM and Process Mining
- Camunda in the insurance industry
- Camunda as AI orchestrator in industrial industry

Andre Strothmann

BPM Consultant

- Process Engineer, Camunda Champion
- *Camunda* and *RPA* in the insurance industry

Who we work for

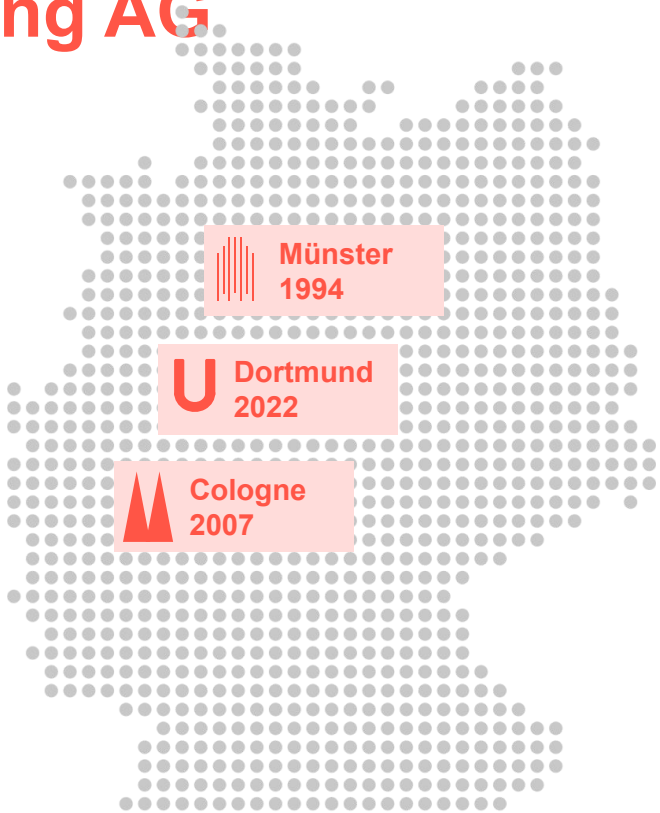
viadee Unternehmensberatung AG

founded 1994 in Muenster, Germany

> 250 consultants

> 40 Mio. EUR revenue p.a.

> 3.500 consulting projects



Areas of competence

- CLOUD ARCHITECTURES & PLATFORMS
- IT SECURITY
- DATA & AI
- BUSINESS ANALYIS
- SOFTWARE DEVELOPMENT & ARCHITECTURE
- ORGANISATIONAL DEVELOPMENT
- BPM & PROCESS MANAGEMENT
- QUALITY ENGINEERING
- PROJECT MANAGEMENT

Agenda

1 Customer Case and Architecture

Service Desk Agent

Customer Case from retail industry

20,000 employees require support from the service desk around the clock.

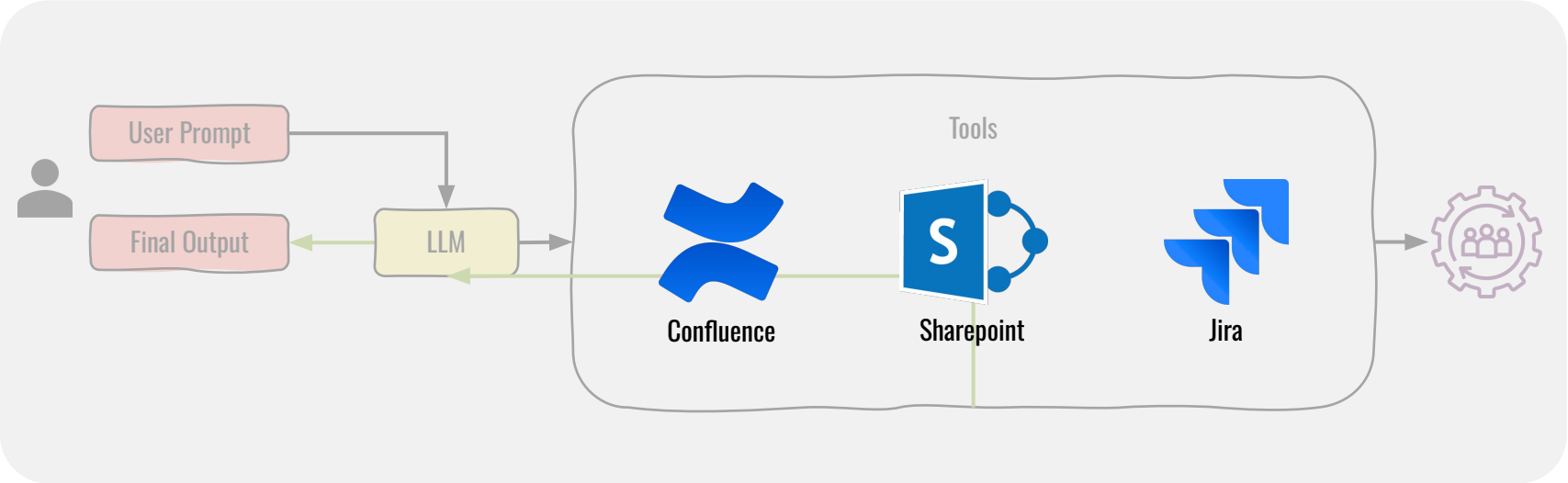
> 2,000 tickets must be processed manually by hotline agents each month.

> *“The keyboard of my laptop is not responding to my typing.”*

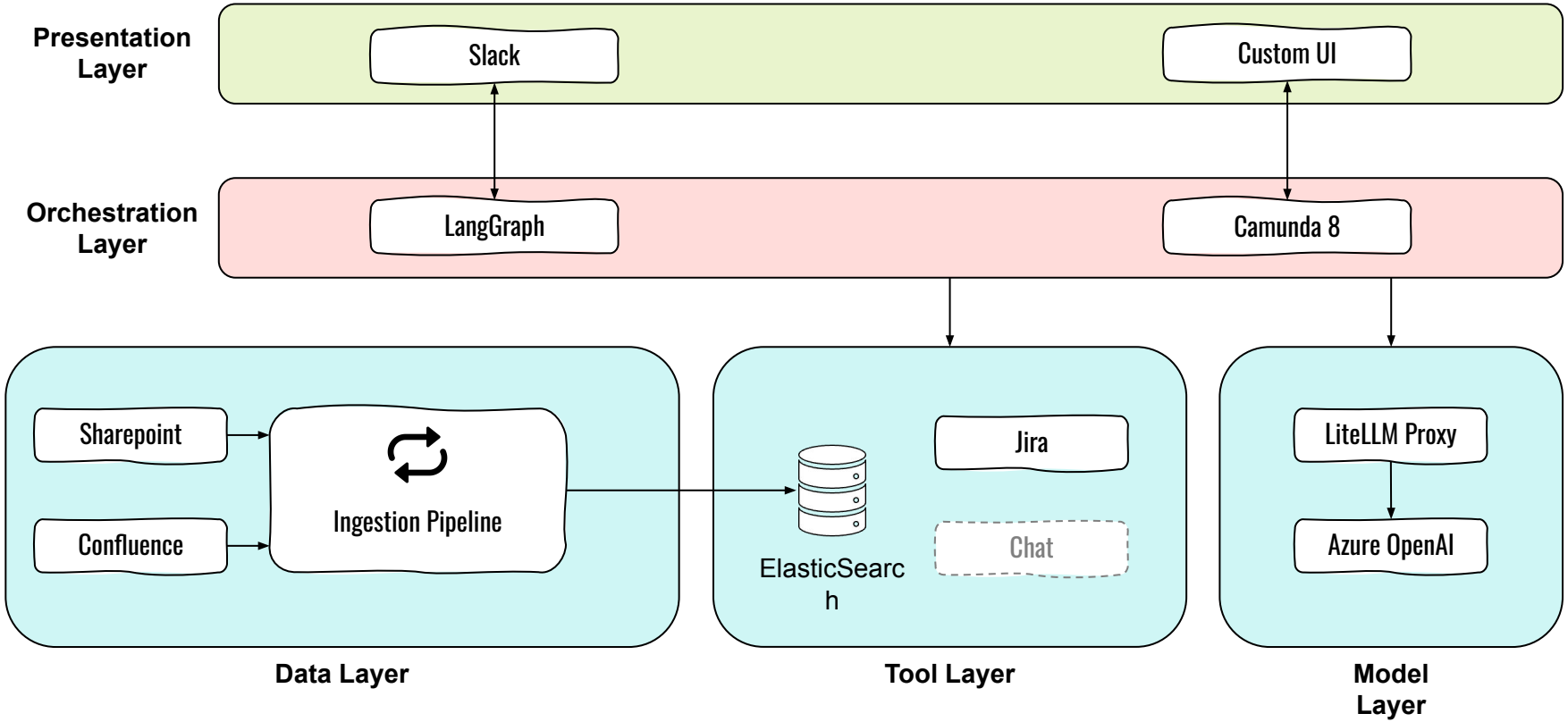
> *“How do I recover deleted emails in Outlook?”*

Customer Case from retail industry

Develop an intelligent chatbot to reduce support tickets and relieve the service desk, enabling employees to resolve issues on their own.



Architecture

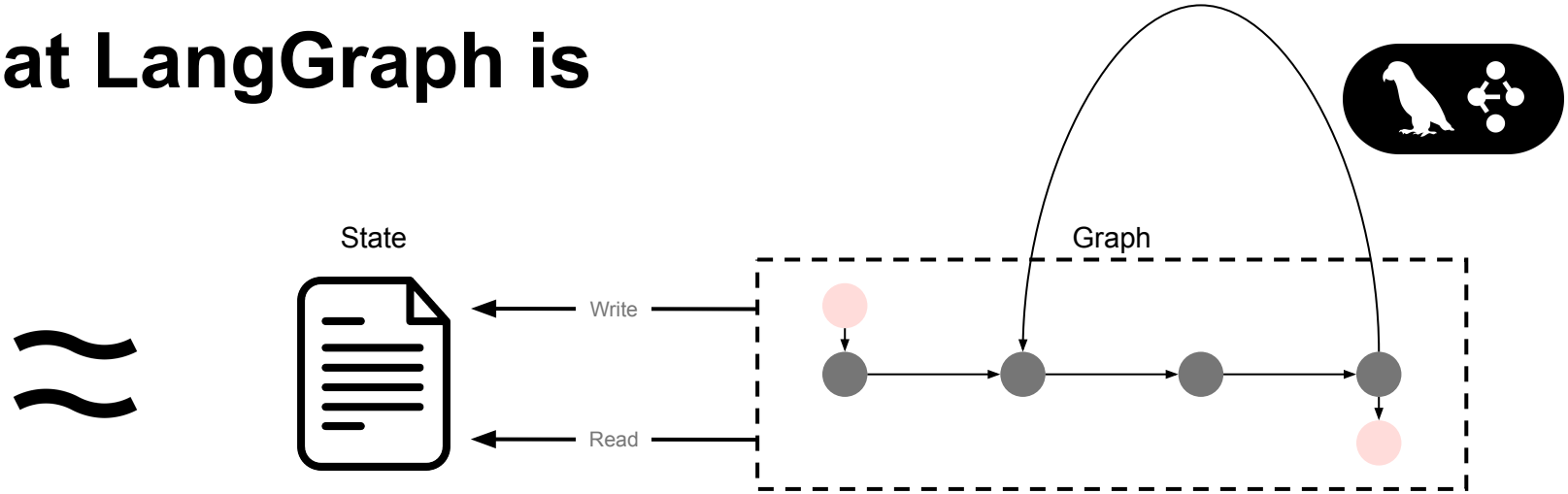


2

Implementation with LangGraph

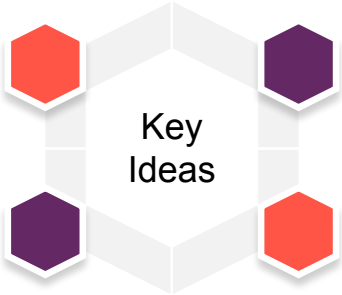
Service Desk Agent

What LangGraph is



Extension to LangChain with graph-based orchestration on top

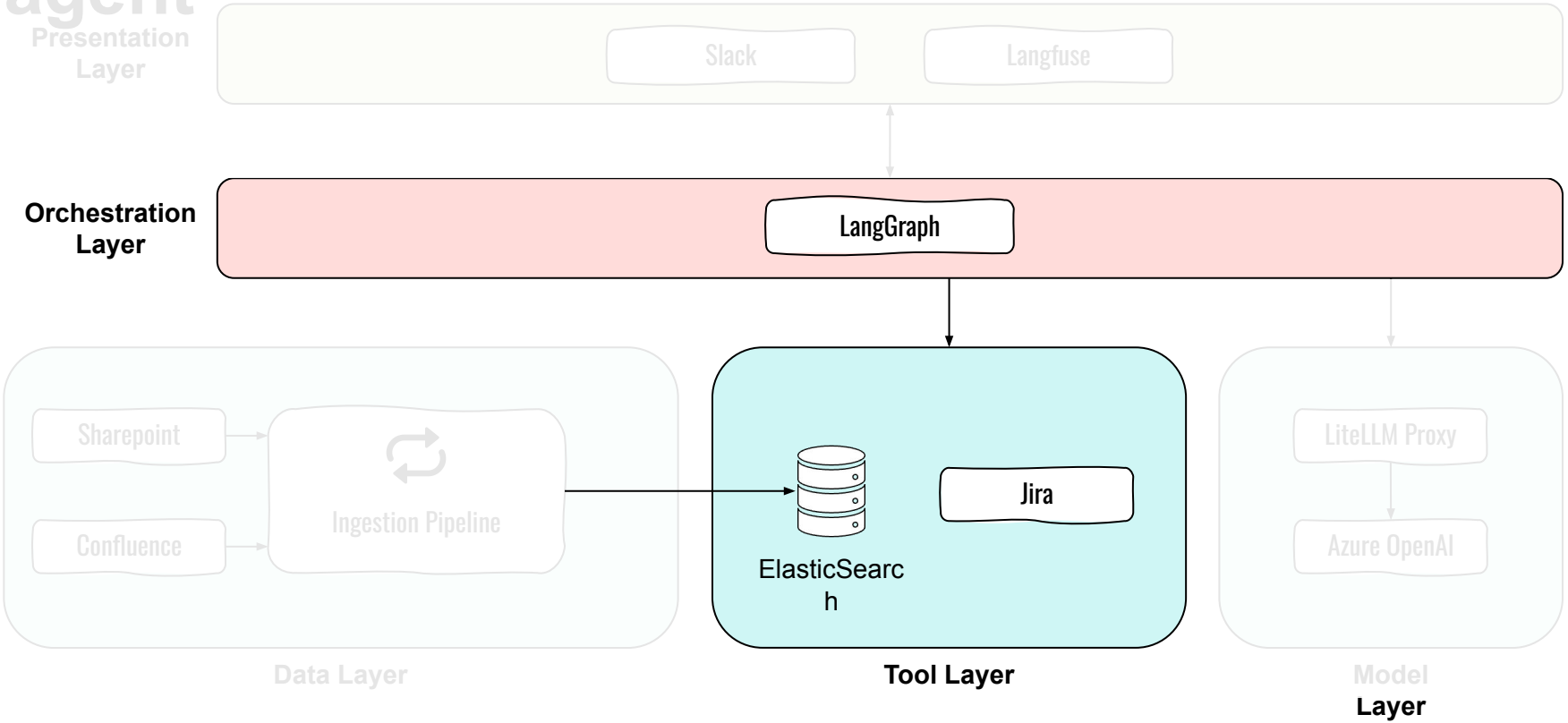
Framework for building AI agents using graph-based workflows



Graph constructs enable if/else, parallel branches, and loops

Open-source, community-driven project that welcomes external contributions

Implementation of service desk agent



Implementation of service desk agent

… in three steps:

1. Definition of available tools
2. Definition of nodes and edges
3. Definition of graph and agent

Implementation of service desk agent

… in three steps:

1 Definition of available tools

Decorator

Description

```python
@tool
def ask_user_for_ticket_creation(request_text):
    """
    This tool can be used to create a Jira ticket for the user at the service desk so that someone can handle the request afterwards.
    """
    summary = get_llm().invoke(f"Generate a short, but descriptive title (do NOT exceed 250 characters) for a jira issue based on the following request: {request_text}").strip('"')
    description = get_llm().invoke(f"Generate a compact description for a jira issue with all details available from the following request: {request_text}")
    ticket = create_jira_ticket(summary, description)
    return f"Successfully created a jira ticket with the following details: {ticket}."

@tool
def query_knowledge_base(request_text):
    """
    Use this tool to search the knowledge database for the AI Agents Inc context.
    """
    query = get_llm().invoke(f"Generate a search query to search for relevant information in the knowledge database based on the following request: {request_text}")
    search_result = answer_query_against_rag(query)
    return f"Searching the knoweldge base has resulted in: {search_result}."
```

Implementation of service desk agent

… in three steps:

2 Definition of nodes and edges

```python
def tool_node(state: dict):
    """Performs the tool call."""
    result = []
    for tool_call in state["messages"][-1].tool_calls:
        tool = tools_by_name[tool_call["name"]]
        observation = tool.invoke(tool_call["args"])
        result.append(ToolMessage(content=observation, tool_call_id=tool_call["id"]))
    return {"messages": result}
```

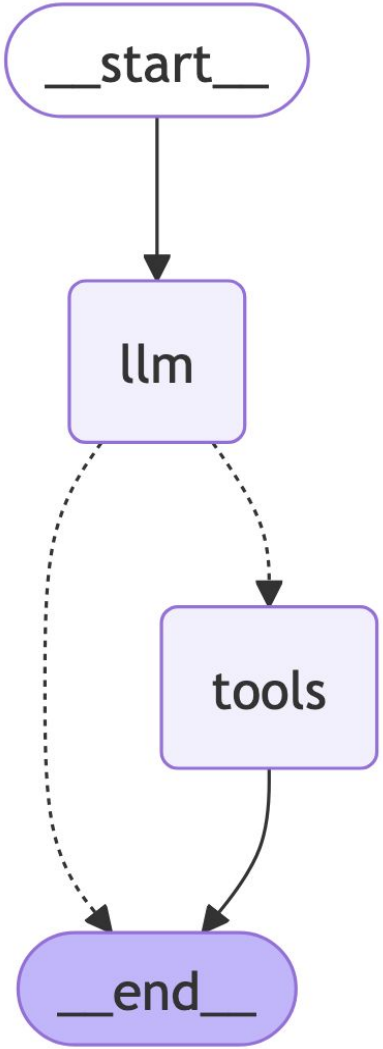
```python
def should_use_tool(state: MessagesState) -> str:
    """
    If the LLM has made a tool call, we are at the tools node.
    If the llm has not made a tool call, we want to escape the loop.
    """
    messages = state["messages"]
    last_message = messages[-1]
    return TOOLS_NODE if last_message.tool_calls else END
```

Implementation of service desk agent

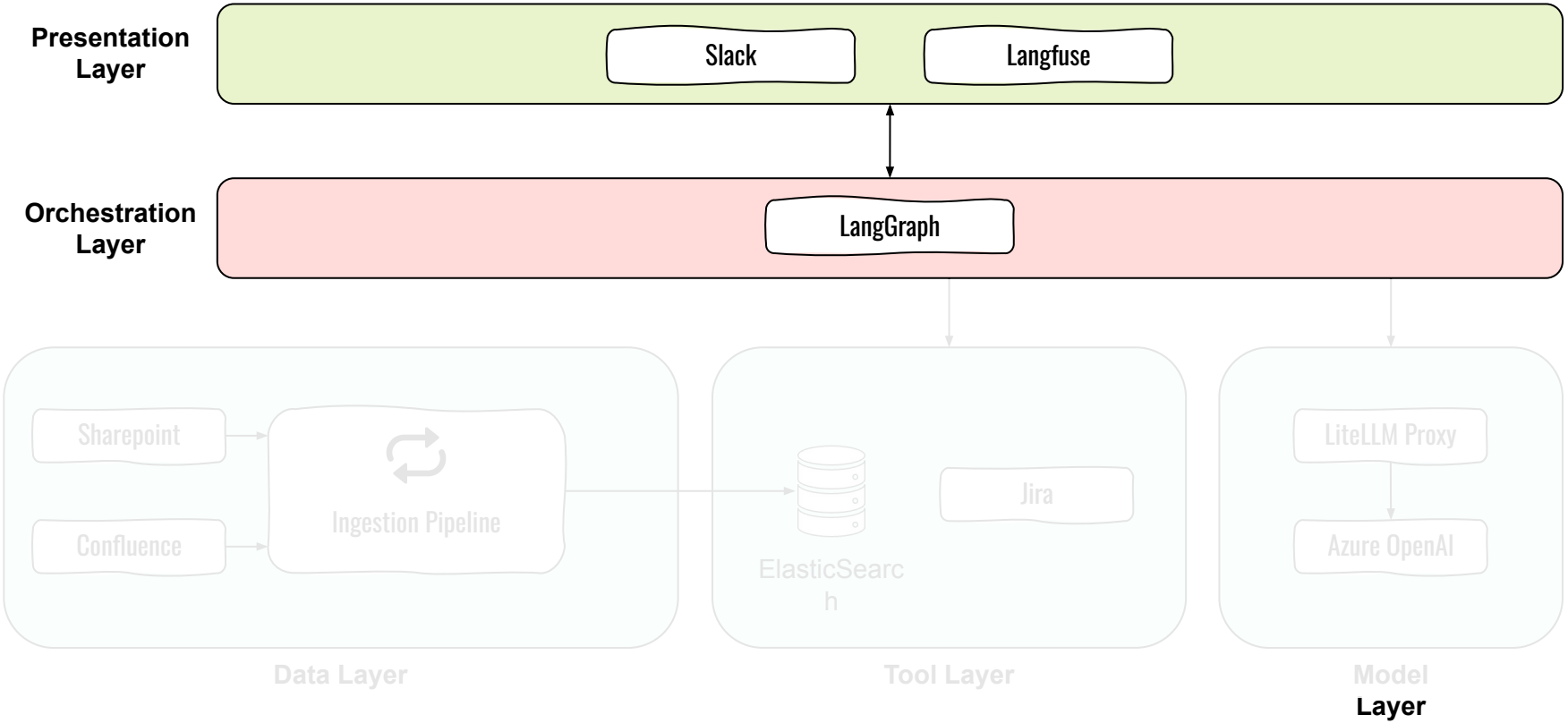
… in three steps:

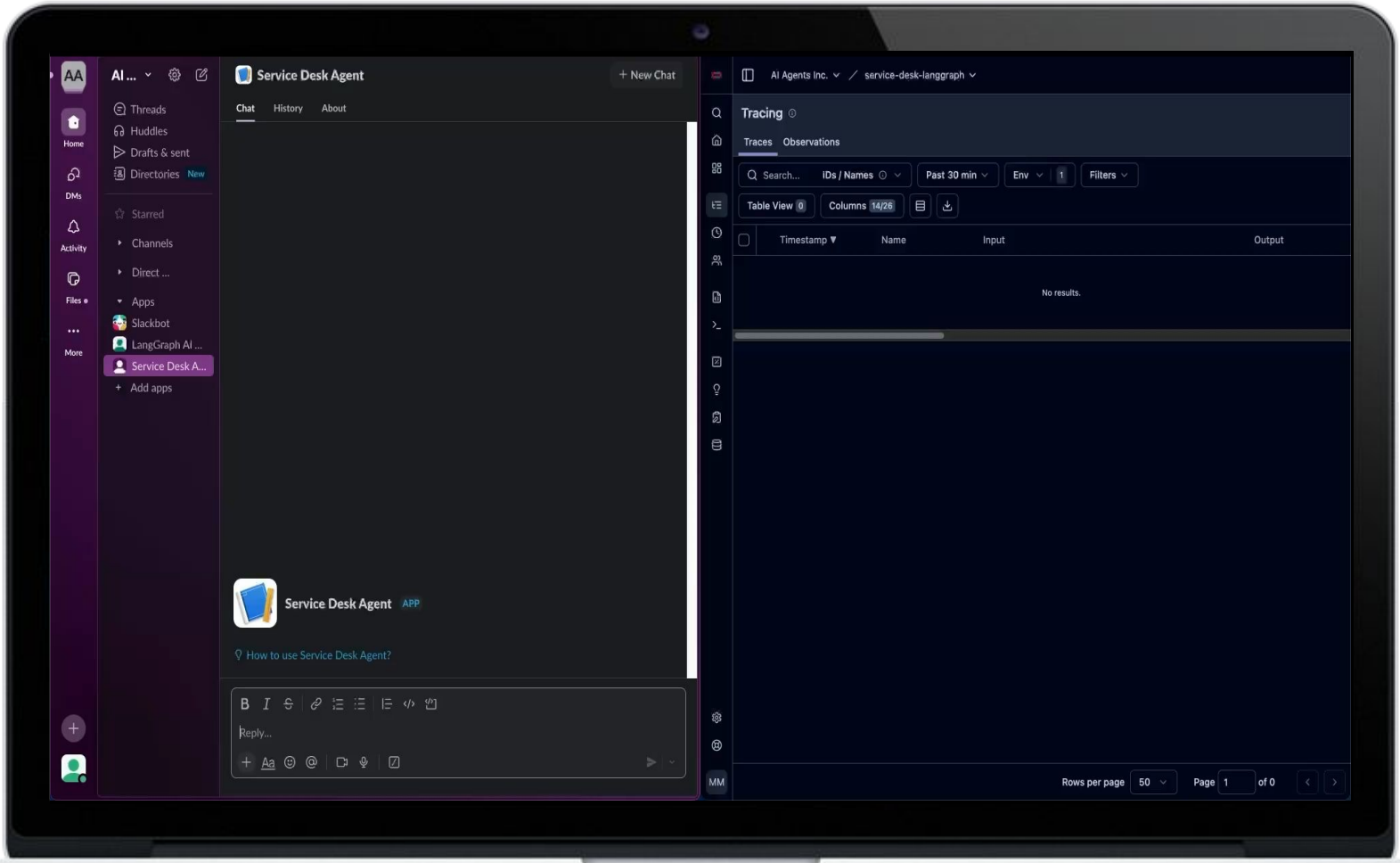
3 Definition of graph and agent

```
agent_builder = StateGraph(MessagesState)
agent_builder.add_node(LLM_NODE, llm_node)
agent_builder.add_node(TOOLS_NODE, tool_node)

# Connect the nodes
agent_builder.add_edge(START, LLM_NODE)
agent_builder.add_conditional_edges(
    LLM_NODE,
    should_use_tool,
    {TOOLS_NODE: TOOLS_NODE, END: END}
)
agent_builder.add_edge(TOOLS_NODE, LLM_NODE)

# Compile the agent
agent = agent_builder.compile().with_config({"callbacks": [langfuse_handler]})
```

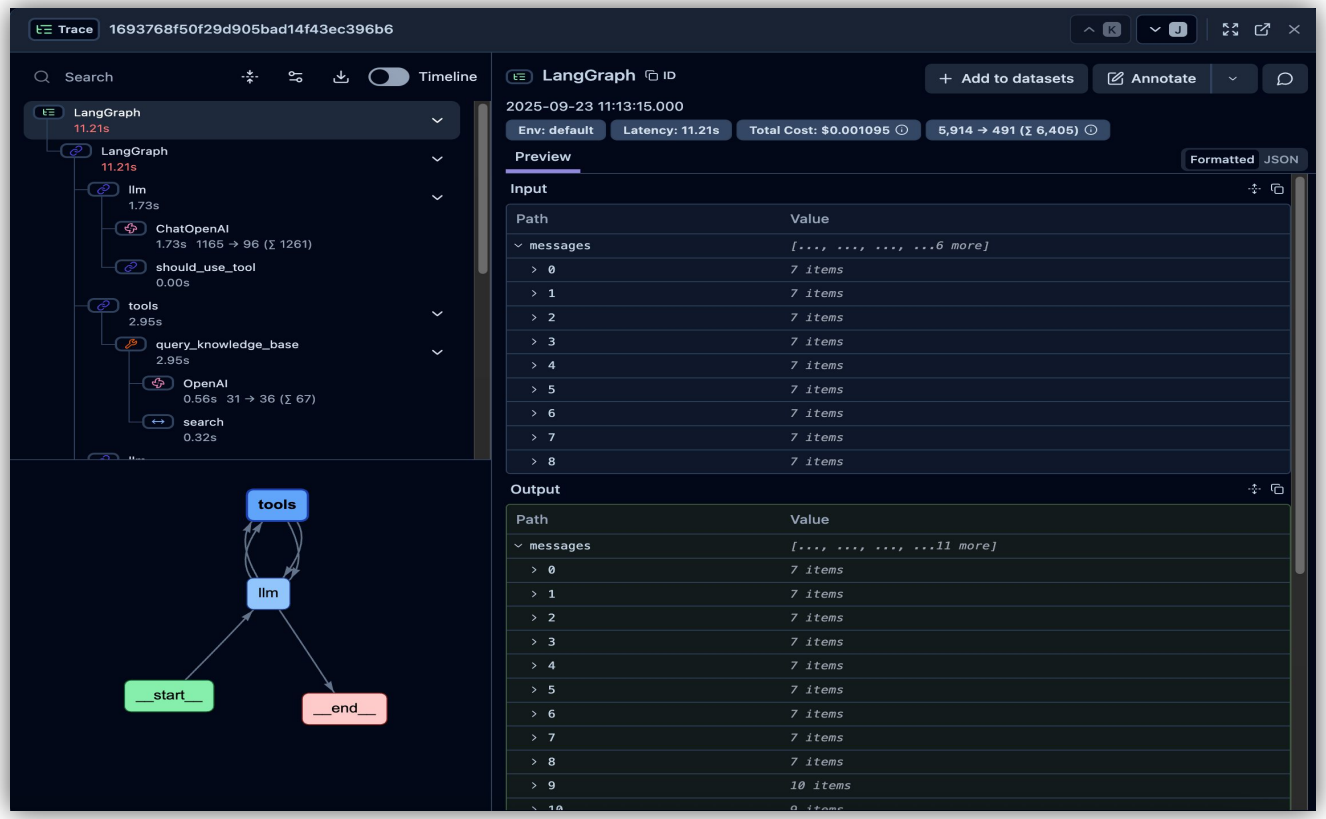


Demo and monitoring




AA
AI ...
Home
DMs
Activity
Files
More
Threads
Huddles
Drafts & sent
Directories New
Starred
Channels
Direct ...
Apps
Slackbot
LangGraph AI ...
Service Desk A...
Add apps
Service Desk Agent
+ New Chat
Chat
History
About
Service Desk Agent APP
How to use Service Desk Agent?
Reply...
AI Agents Inc.
service-desk-langgraph
Tracing
Traces
Observations
Search...
IDs / Names
Past 30 min
Env
1
Filters
Table View 0
Columns 14/26
Timestamp
Name
Input
Output
No results.
MM
Rows per page
50
Page
1
of 0

Monitoring with Langfuse

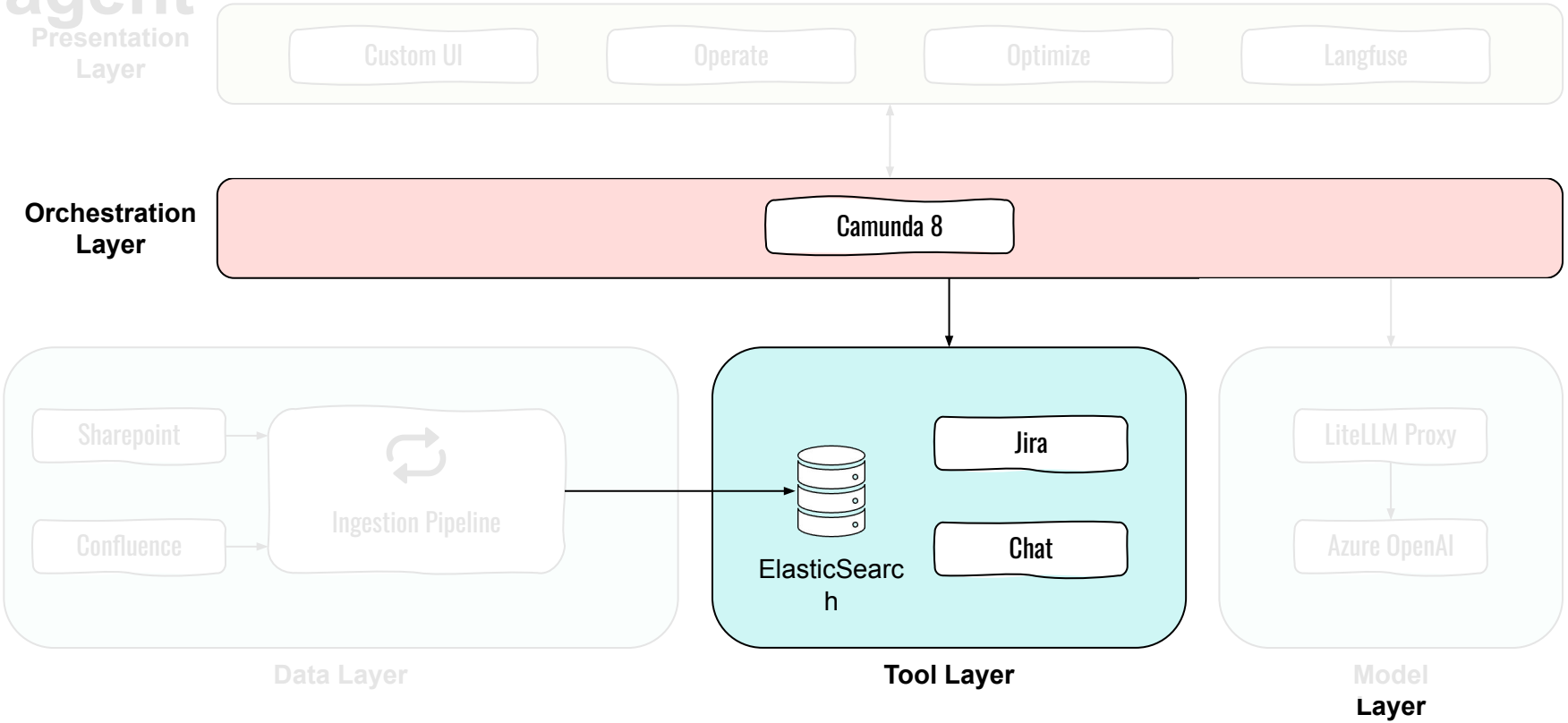


3

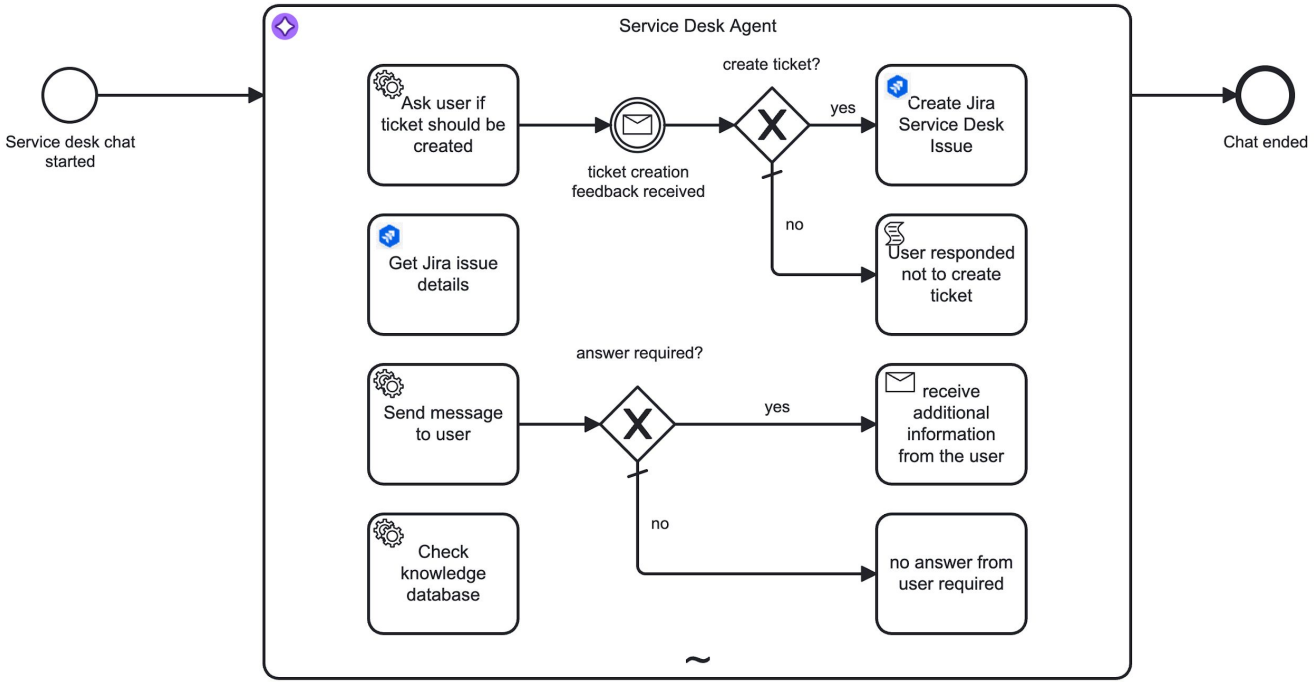
Implementation with Camunda 8

Service Desk Agent

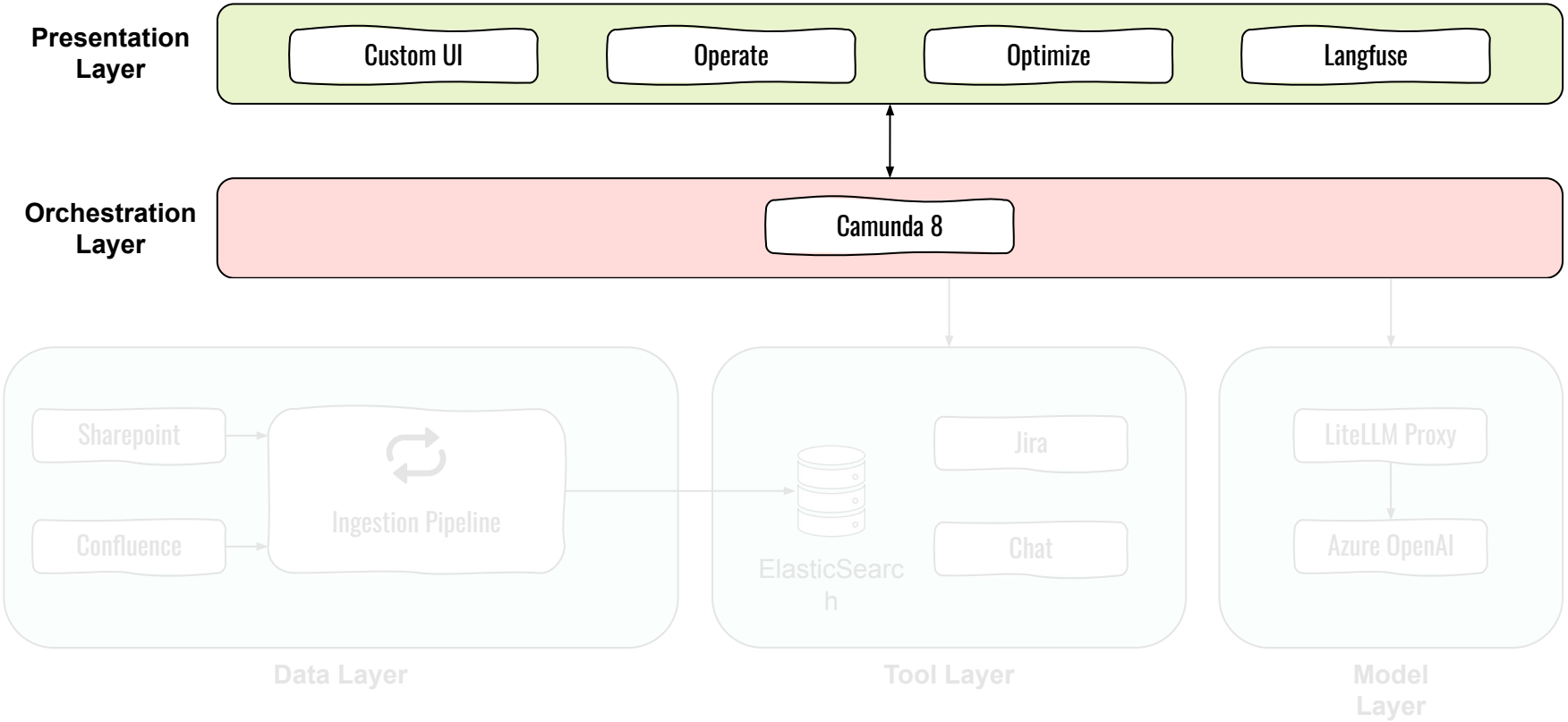
Implementation of service desk agent

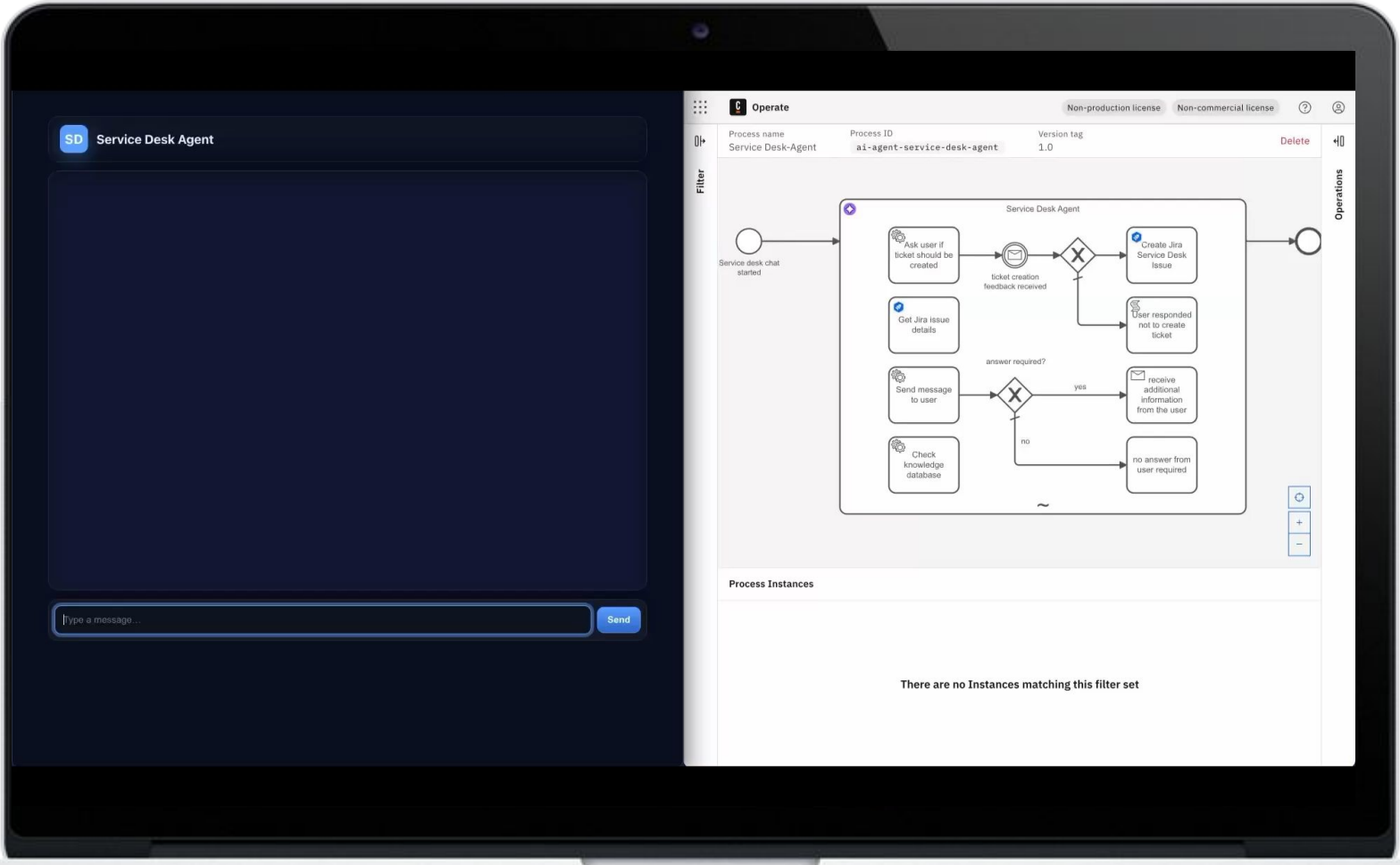


AI Agent in Camunda



Demo and monitoring




SD
Service Desk Agent
Type a message...
Send
Operate
Non-production license
Non-commercial license
Filter
Process name
Service Desk-Agent
Process ID
ai-agent-service-desk-agent
Version tag
1.0
Delete
Operations
Service Desk Agent
Service desk chat started
Ask user if ticket should be created
ticket creation feedback received
Create Jira Service Desk Issue
User responded not to create ticket
Get Jira issue details
answer required?
Send message to user
yes
receive additional information from the user
no
Check knowledge database
no answer from user required
Process Instances
There are no Instances matching this filter set

Operational transparency

Operate

- Complete flow of each agent interaction visible
- Tool sub-step level (deterministic steps)

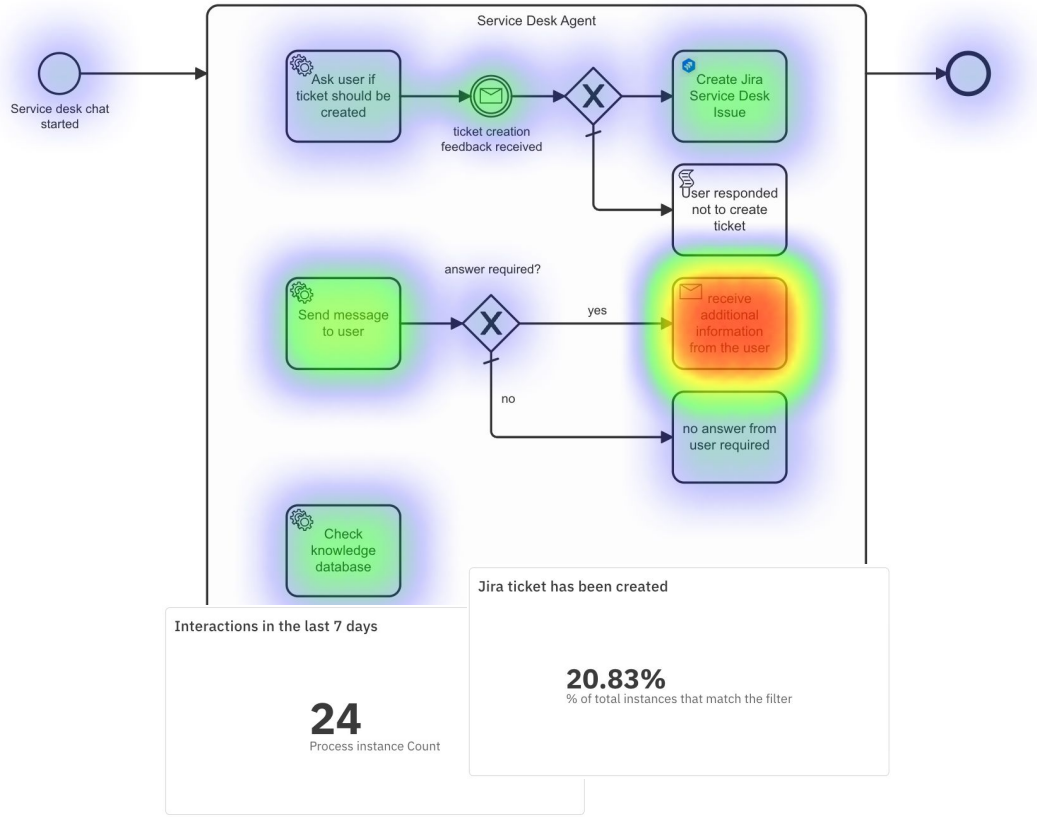
Separate ai agent pattern

- Service Desk-Agent
  - Service desk chat started
  - Gateway_0a2lv5m
  - Service Desk Agent
  - use tools?
  - agentTools (Multi Instance)
    - agentTools
      - Send message to user
      - answer required?
      - receive additional information from the user
  - Gateway_0a2lv5m
  - Service Desk Agent
  - use tools?
  - agentTools (Multi Instance)
    - agentTools
      - Send message to user
      - answer required?
      - receive additional information from the user
  - Gateway_0a2lv5m
  - Service Desk Agent
  - use tools?
  - agentTools (Multi Instance)
    - agentTools
      - Check knowledge database

Operational transparency

Optimize

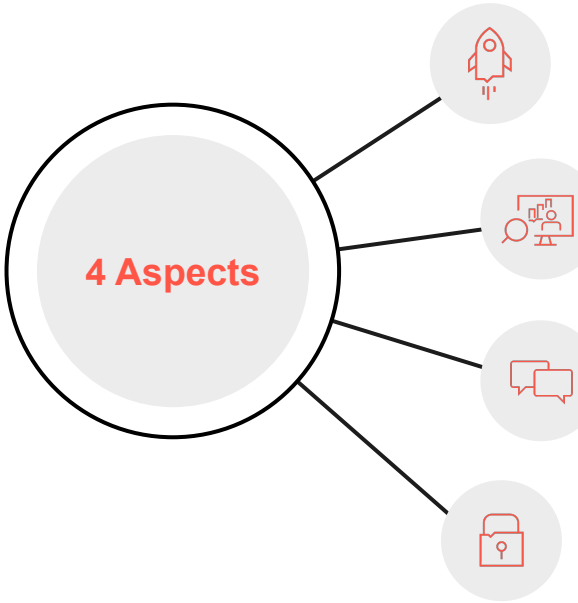
- Process metrics
- Heatmaps of tool usage



4 Comparison

LangGraph vs. Camunda 8

Key Aspects for Comparison



Agent Architecture & Control Flow
How is the agent behavior approached?

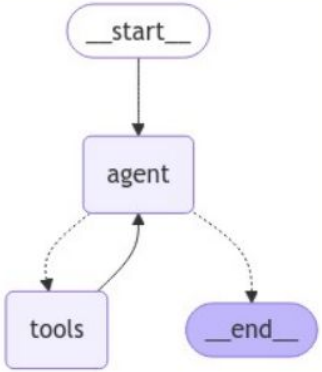
Tracing & Observability
How is monitoring and transparency achieved?

Conversational Agents
How are natural language interactions supported?

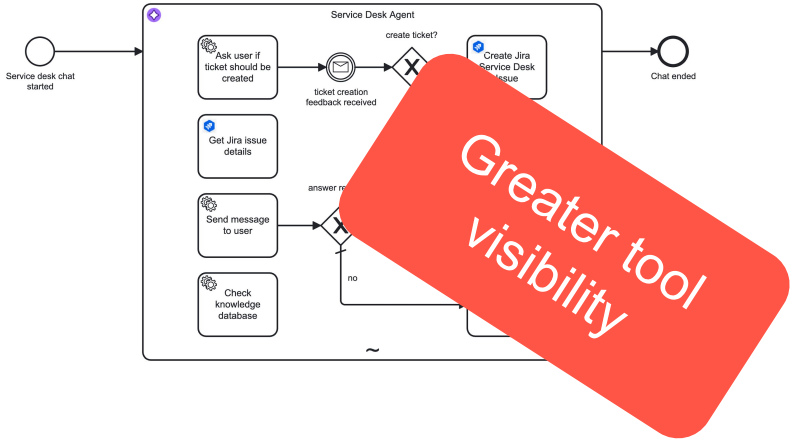
Control Mechanisms
What options exist for governance, rules, and decision control?

Agent Architecture & Control Flow

Code-first modular agent architecture that allows dynamic control flow

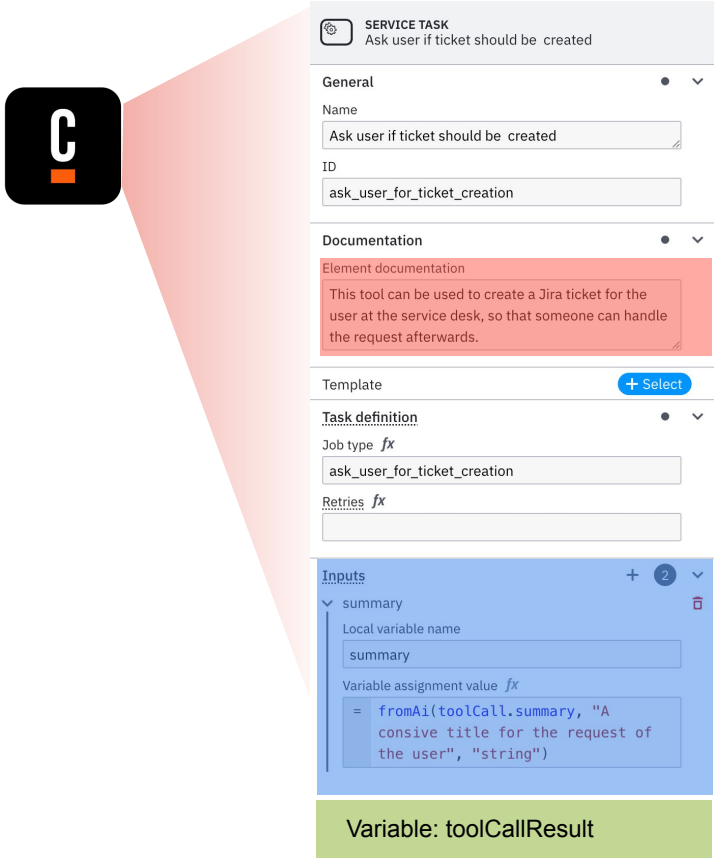


Process-first architecture with static control flow embedded in ad-hoc subprocess



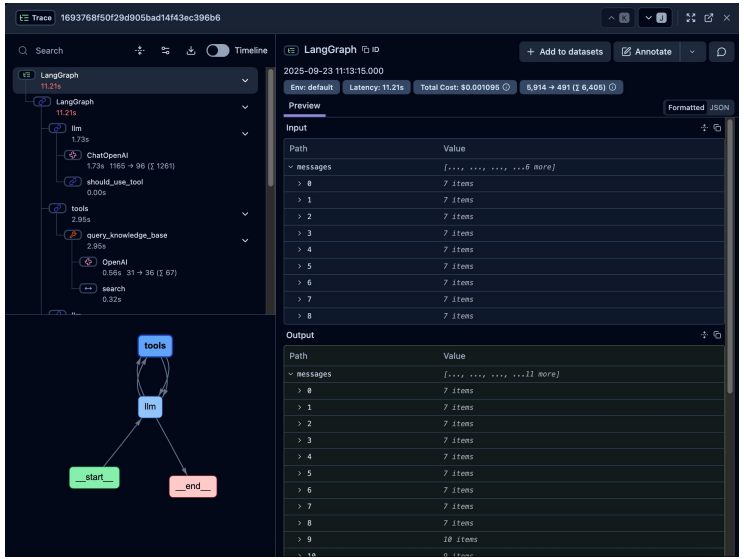
Example: Tool definition

```python
@tool
def ask_user_for_ticket_creation(request_text):
    """
    This tool can be used to create a Jira ticket for the user at the service desk,
    """
    summary = get_llm().invoke(f"Generate a short, but descriptive title (do NOT exc
    description = get_llm().invoke(f"Generate a compact description for a jira issue
    ticket = create_jira_ticket(summary, description)
    return f"Successfully created a jira ticket with the following details: {ticket}
```

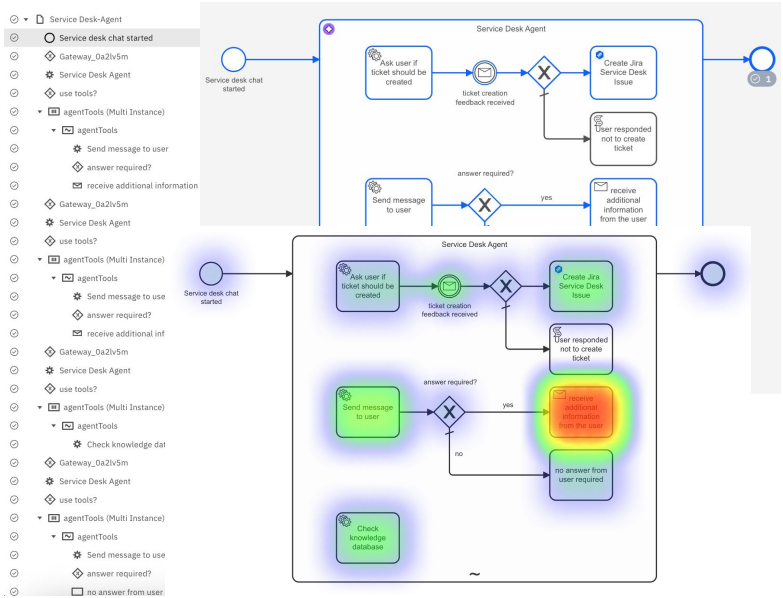


Variable: toolCallResult

Tracing & Observability



Graph-Level transparency



Process-level transparency

LLM observability in Camunda 8

Trace f85f8a0f-952b-4f37-a7c2-b3bfa4365a3f

Search | Timeline

Hotline Agent Camunda 8
3m 10s
Conciseness: 0.70

litellm:camunda8
1.30s 1196 → 33 (∑ 1229)
litellm:camunda8
0.73s 1251 → 55 (∑ 1306)
litellm:camunda8
0.81s 1330 → 25 (∑ 1355)
litellm:camunda8
0.61s 7 → 0 (∑ 7)
litellm:camunda8
2.61s 2100 → 122 (∑ 2222)
litellm:camunda8
2.09s 2256 → 80 (∑ 2336)
litellm:camunda8
1.62s 2301 → 144 (∑ 2445)
litellm:camunda8
0.87s 2383 → 85 (∑ 2468)
litellm:camunda8
1.05s 1712 → 76 (∑ 1788)
litellm:camunda8
0.74s 1649 → 54 (∑ 1703)

litellm:camunda8 ID | + Add to datasets | Annotate | Playground

2025-09-05 08:34:40.348

Latency: 0.81s | Time to first token: 0.81s | Env: default | $0.000236
1330 prompt → 25 completion (∑ 1355) | gpt-4o-mini | temperature: 0.5 | stream: false | max_retries: 0
extra_body: {} | system_fingerprint: fp_efad92c60b

Preview | Formatted | JSON

- ATP Web Portal: https://atp-web.ewer.local/

The answer must always be generated in the english language.

Show 4 more ...

Tool

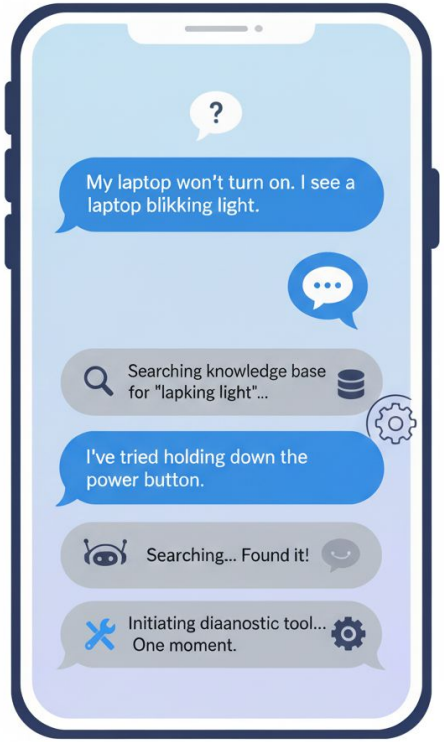
the feedback from the user was: The esc key is not owkring

Path	Value
∨ json	1 items
tool_call_id	"call_j3iA7GvJtW3EiZ3VYTEpaZxF"

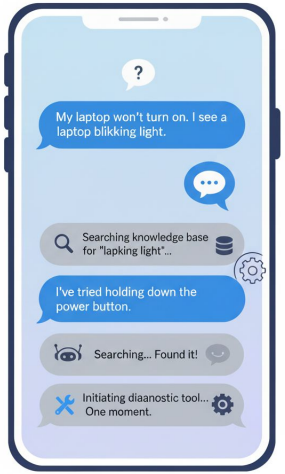
Assistant

Path	Value
∨ tool_calls	[{"function": ...}]
∨ 0	3 items
∨ function	2 items
∨ arguments	1 items
queryKnowledgeBase	"keyboard issues, ESC key not working"
name	"query_knowledge_database"
id	"call_GgELfcvo7LAf00PXs6wxCzJo"
type	"function"
function_call	null
annotations	empty list

Conversational Agents



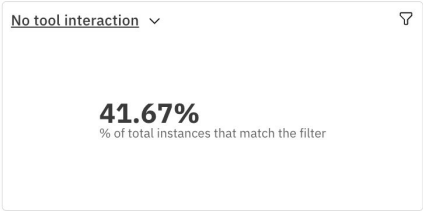
Conversational Agents



- Natively conversational
- Natural language is the control interface

- BPMN not natively dialog-oriented
- Agent can become conversational by using LLM-based tools
- Less natural for free-flowing, dynamic dialogue compared to LangGraph

Prompting important, but also not reliable



Control Mechanisms

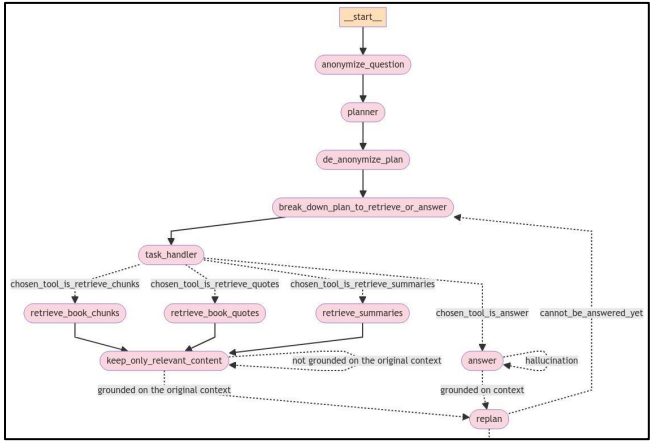
- Governed by schema validation, tool restrictions, and moderation layers
- Allows dynamic guardrails during execution, maintaining control
- More flexible, but requires custom policy design and guidance

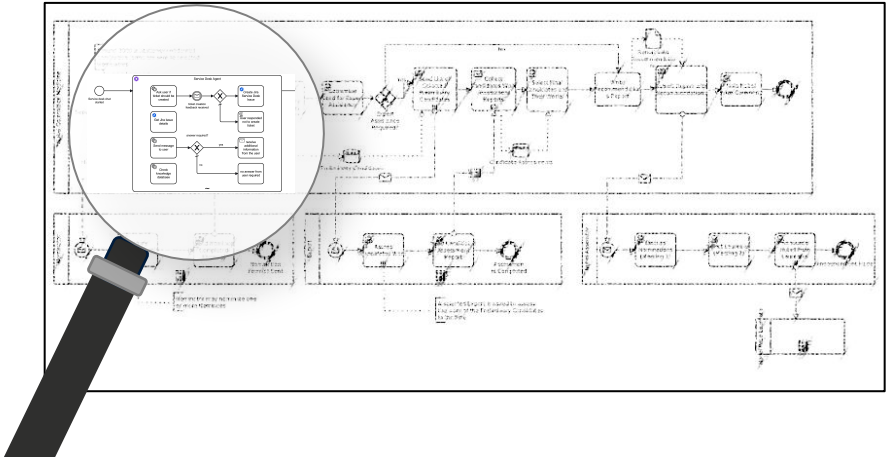
- Uses BPMN and DMN for a structured flow of control during execution
- Strong support for process-level policies, approvals, and roles
- Easy to integrate with enterprise governance and compliance systems

Both platforms support guardrails - but in different ways

Comparing Apples & Oranges?

VS

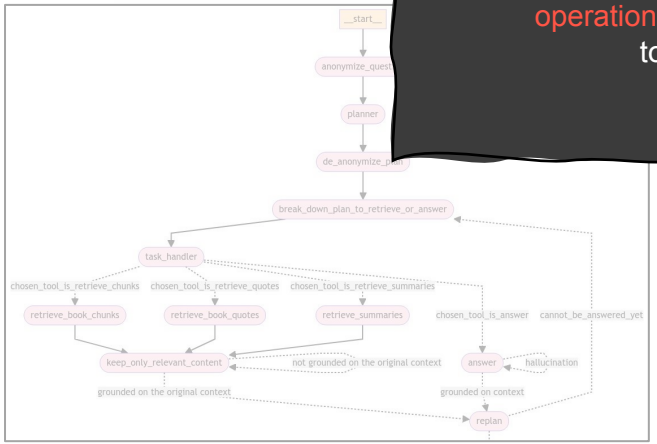


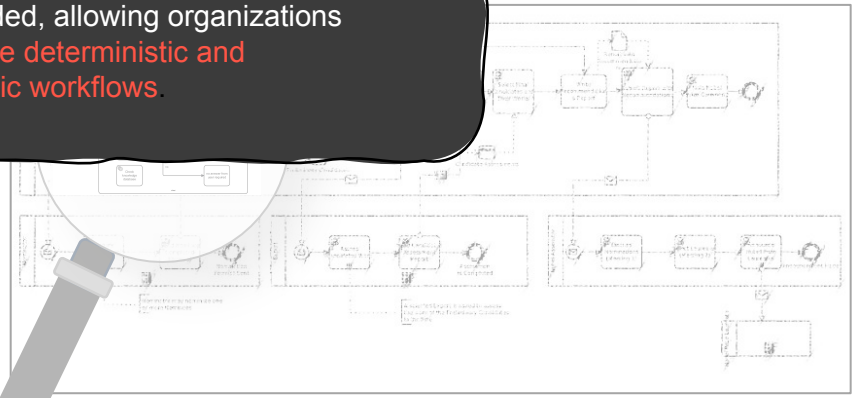


Comparing Apples & Oranges?

VS

LangGraph is specialized to build agents, whereas Camunda 8 is designed to automate end-to-end business processes and operationalize agents as needed, allowing organizations to seamlessly combine deterministic and non-deterministic workflows.





IT COMES DOWN TO USE CASES!

viadee

Thank you!
Questions?

Andre Strothmann
Andre.Strothmann@viadee.de

viadee Unternehmensberatung AG
Anton-Bruchausen-Str. 8
48147 Münster
Tel: +49 251 77777 377
www.viadee.de

Mario Micudaj
Mario.Micudaj@viadee.de

viadee Unternehmensberatung AG
Anton-Bruchausen-Str. 8
48147 Münster
Tel: +49 251 77777 138
www.viadee.de

viadee

entdecken entwickeln
entfalten

viadee Unternehmensberatung AG
viadee.de

Preparing the knowledge base

1 Extract

Extract and prepare data from Confluence and Sharepoint.

Chunk 2

Segment the extracted content into chunks.

3 Embed

Create embeddings for each chunk and insert into vector store including metadata..

Insert 4

1st Chunk

Overview

Device as a Service (DaaS) is a modern IT procurement and management model where hardware devices (e.g., laptops, desktops, tablets, smartphones) are offered as a subscription bundle together with lifecycle services. Instead of purchasing devices outright, the company pays a predictable monthly or annual fee for the hardware, software, and related support services.

DaaS shifts device ownership and lifecycle responsibility from the company to the service provider, helping reduce complexity, costs, and operational risks.

2nd Chunk

Key Characteristics

- **Subscription-based model**

  Devices and services are bundled into a recurring fee (OPEX instead of CAPEX).
- **Lifecycle management**

  Covers procurement, deployment, maintenance, and retirement/recycling of devices.
- **Scalability**

  Devices can be added, upgraded, or removed easily based on business needs.
- **Vendor responsibility**

  Service providers handle logistics, support, and updates.
- **Analytics & insights**

  Many providers include usage data, performance monitoring, and predictive maintenance.

3rd Chunk

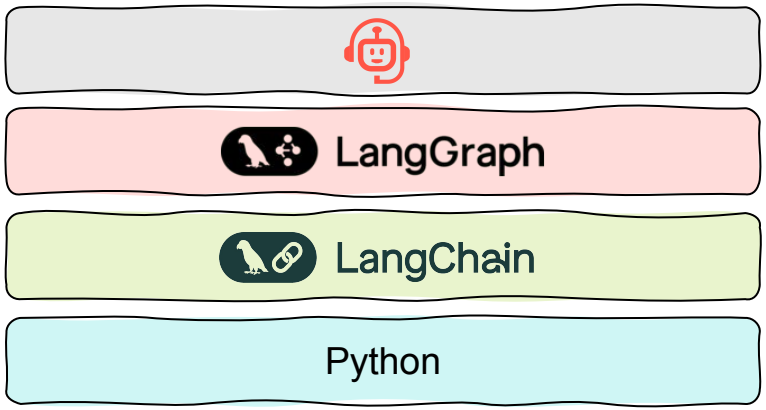
Benefits for the Company

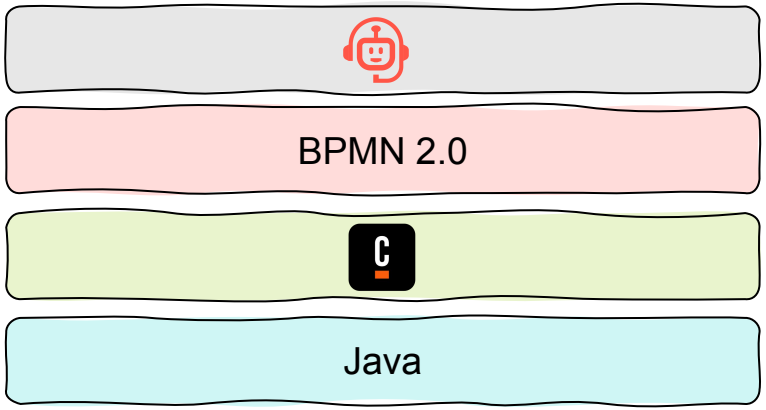
- **Cost predictability**: No large upfront investment, easier budgeting.
- **Flexibility**: Scale device fleet up or down quickly.
- **Efficiency**: Faster onboarding/offboarding with pre-configured devices.
- **Innovation**: Regular refresh cycles keep devices up to date with the latest technology.
- **Sustainability**: Providers manage recycling and end-of-life responsibly.
- **Focus on core business**: IT resources can concentrate on strategic tasks instead of device management.

Technology Stack

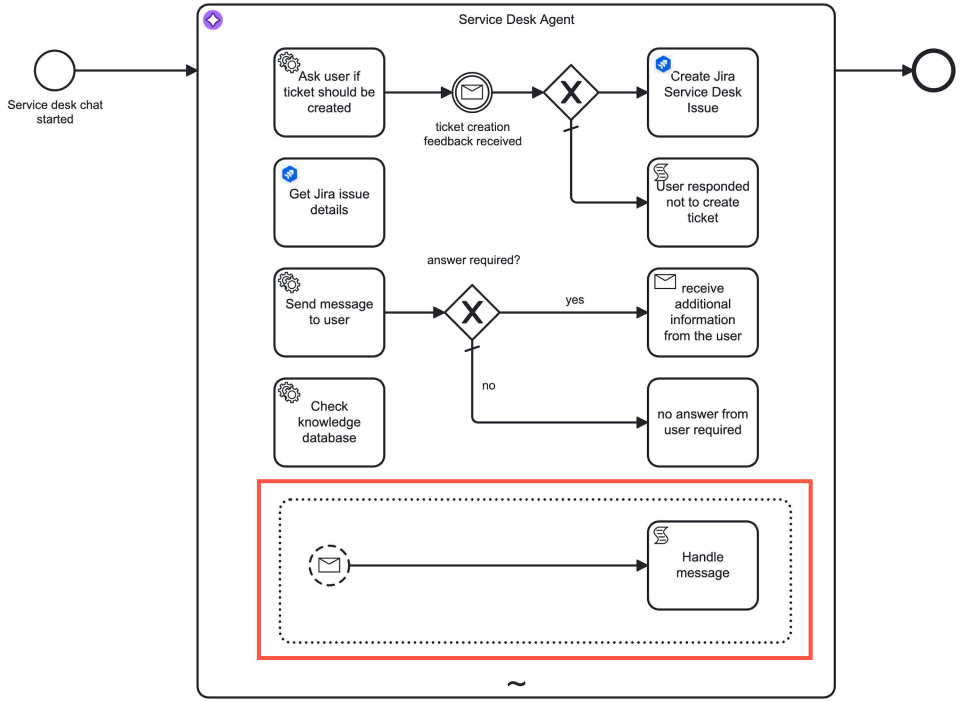
- low-level orchestration framework for building ai agents
- Code-oriented

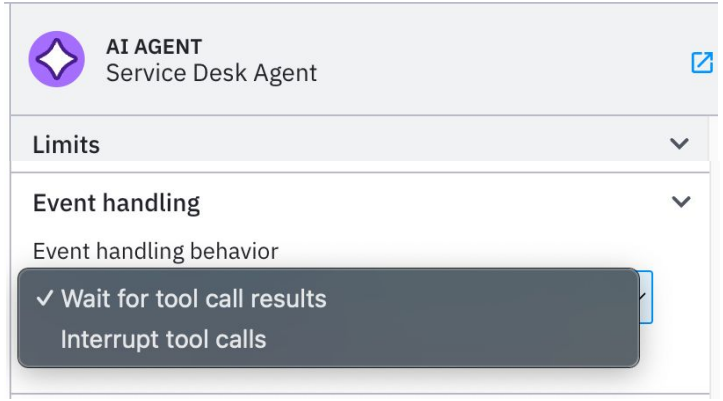
- AI Agent connector based on LangChain4j
- Model-oriented





Event subprocess in AI Agent





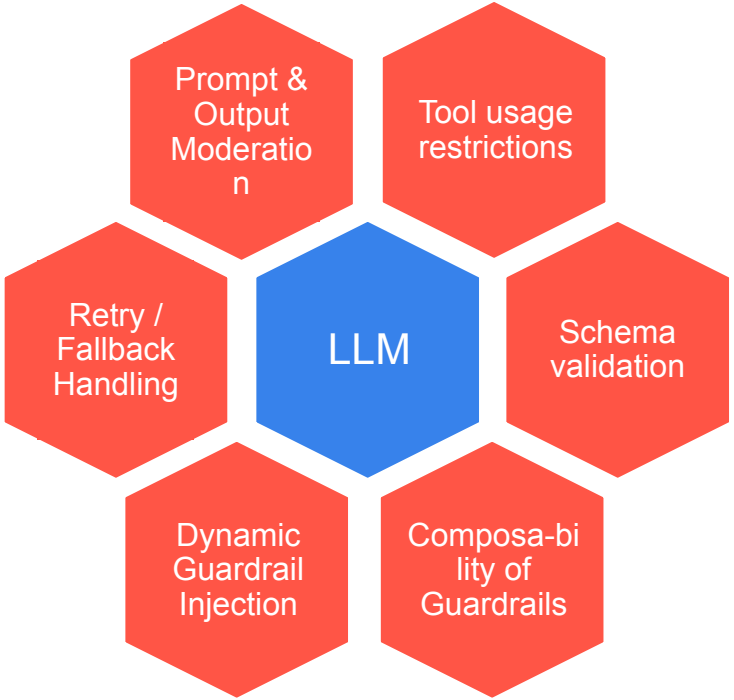
Support for „interrupting“ message input

Guardrails in LLM Model provider

LLM guardrails

Guardrails on tool output

Guardrails on graph edges



LLM guardrails of model provider

Limited by modelling possibilities