Oracle TechTalk:

Agent Spec

Run AI Agents Anywhere with Open Agent Specification

Louis Faucon, Database AI Research

13th November 2025

Agenda

- **Background** on LLMs and AI Agents
- Introduction to **Agent Spec**
- **Demo**: Structured Generation
- **Demo**: ReAct agent with Tools
- **Demo**: ReAct agent with MCP

Background

Background – from LLMs to conversational agents

LLMs purely complete missing text; Conversational agents helpfully reply to requests;

Multiple training phases are necessary to produce helpful LLMs for Agents:

- Pretraining: Using very large corpus of text data from the internet
- Prompting: Using System/Agent/User message to produce a conversation.
- Finetuning: Training the model on dedicated data for following instructions
- Alignment: with the preferences of the LLMs developers using Reinforcement Learning from Human Feedbacks

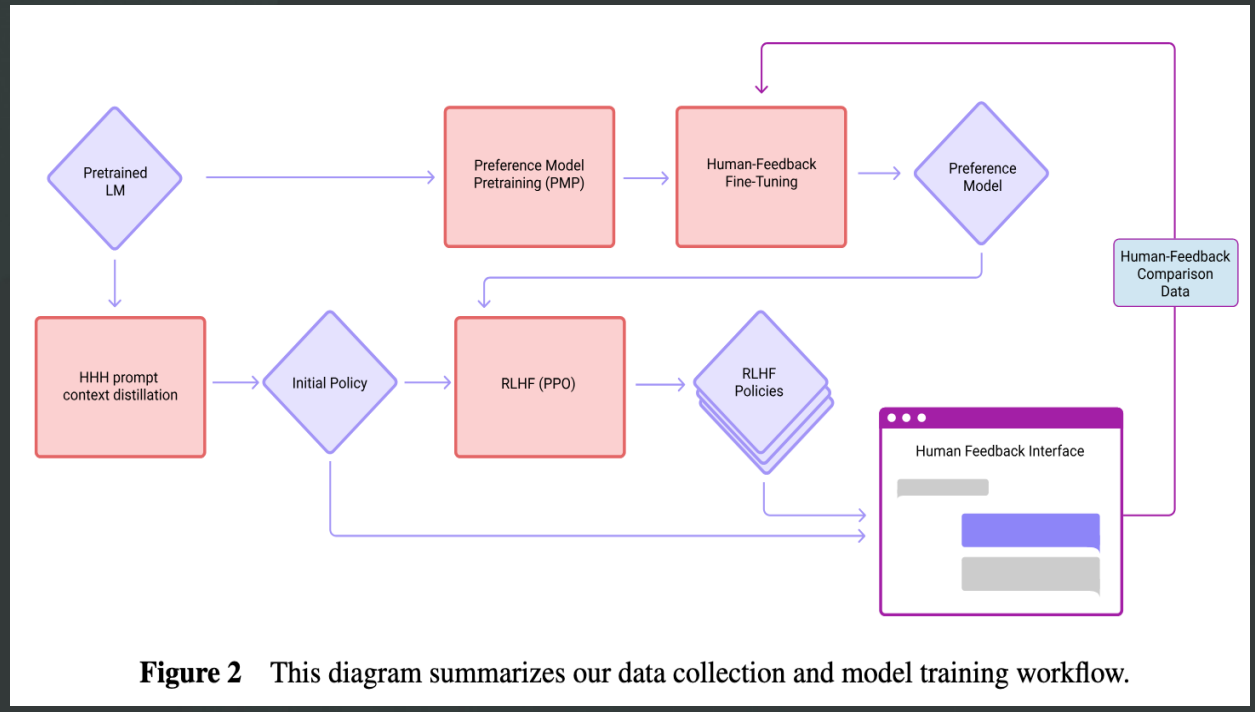


Figure 2 This diagram summarizes our data collection and model training workflow.

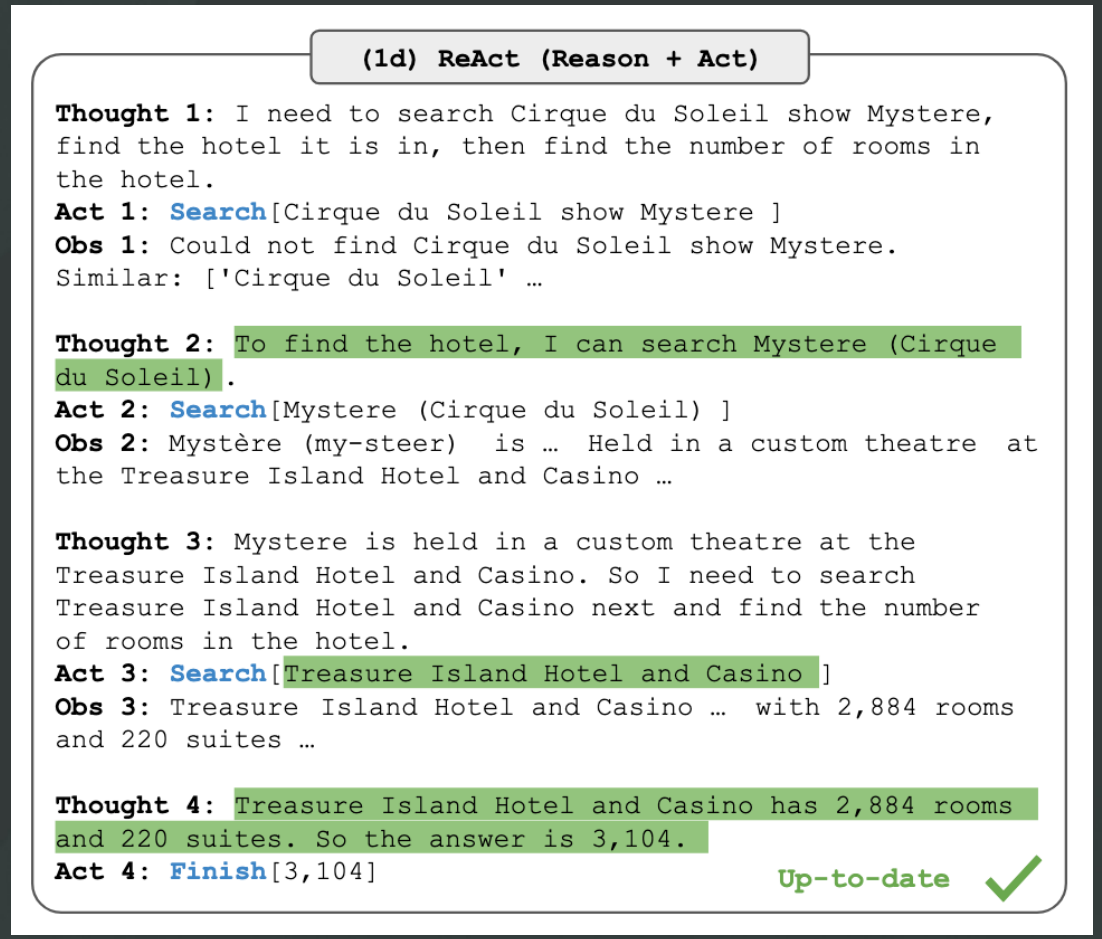
Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback, Anthropic https://arxiv.org/abs/2204.05862

Background – ReAct Agents

Connecting tools to agents to take actions in the real world

LLMs can be made to decide to invoke external tools:

- System prompt: Contains a description of the tools and how to invoke them
- When receiving a user request, the LLM can then generate either a reply to the user or a Tool Call following a specific format (e.g. JSON)
- After the model generates a Tool Call, the tool is executed and its result appended to the conversation of the Agent as an observation

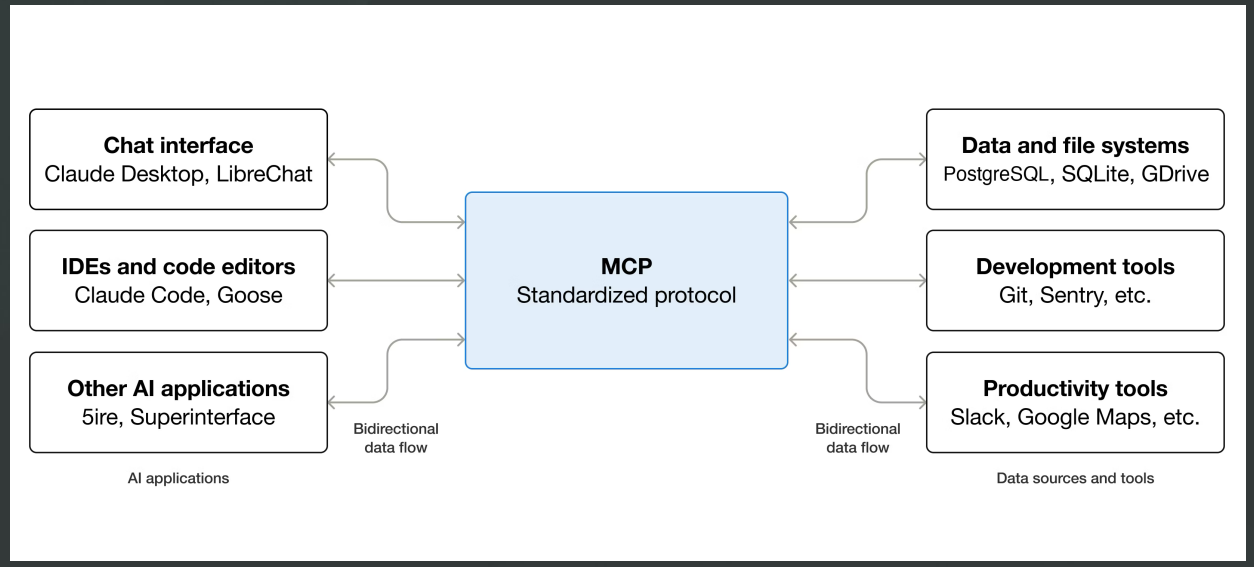


ReAct: Synergizing Reasoning and Acting in Language Models
https://arxiv.org/abs/2210.03629

Background – Model Context Protocol

Standardizing how applications provide tools and context to LLM Agents.

- Open Source standard supported by many systems and platforms
- Simplifies connecting your agents to external services (e.g. web APIs, search engines, …)
- Provides tools, data sources, and specialized prompts
- MCP servers can either be deployed by application providers or can be run locally by users directly



What is MCP?
https://modelcontextprotocol.io/docs/getting-started/intro

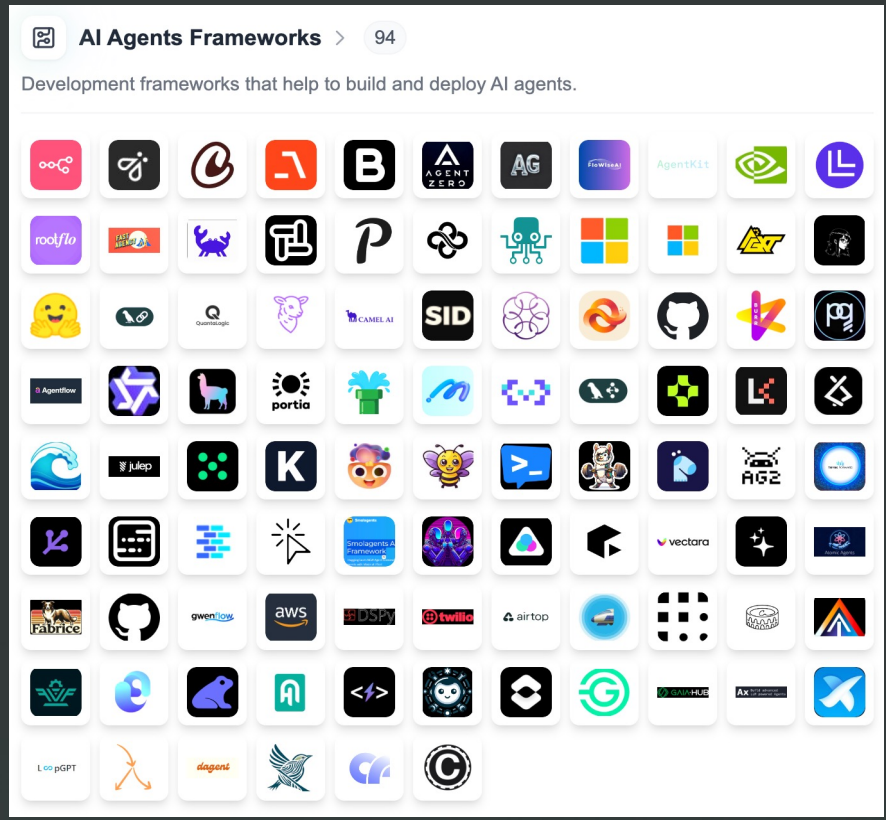
Open Agent Specs

Open Agent Specs – Motivation & Context

Proliferation Without Reusability

Proliferation of Agents frameworks:

- 100+ open-source and commercial frameworks
- No common abstractions for defining agents, tools, and flows
- Reusing components (e.g., memory modules, planners, toolchains) across frameworks is manual and error-prone



Learnings from Deep Learning & ONNX unification

- Early ML frameworks were siloed: Caffe, Torch, Keras, TensorFlow…
- Open Neural Network Exchange (ONNX) unified model representation and enabled reusability of models
- Resulted in a shared ecosystem of reusable models and tooling

➔ Open Agent Spec brings a shared foundation that would enable for AI Agents what ONNX did for Deep Learning

Open Agent Specs - A Unified Agent Specification

What is Open Agent Specification (**Agent Spec**)?

"A **framework-agnostic configuration language** for defining AI Agents & Flows."

Frameworks-agnostic configurations

Agent Spec is a configuration language **independent** from any specific agent framework.

Runtime adapters can be implemented to support Agent Spec **in any agent framework.**

Reproducibility & exchangeability

Every agent defined with Agent Spec can be **executed and evaluated** consistently.

Standardized schemas increase **reproducibility, and exchangeability** across environments.

Modularity & Reusability

Agent Spec provides **pre-built components** and **composition primitives** that make it easy to build, extend, and reuse agents across applications.

Open Agent Specs – Representation and Runtime adapters

Enabling deployment to the right platform

Agent Spec is purely a representation of an agent or agentic workflow

A runtime adapter makes Agent Spec components executable by implementing the behavioral logic defined in the representation

Developers can prototype and develop on one platform (e.g. langgraph) deploy on another

Agent Spec Representation (e.g. RAG Agent)

```
{
  "component_type": "Agent",
  "name": "adaptive_expert_agent",
  "inputs": [{"title": "domain_of_expertise",
…}],
  "outputs": [],
  "llm_config": {
    "component_type": "VllmConfig",
    "name": "Llama 3.1 8B instruct",
    "url": "http://llama.deployment.url/v1",
    "model_id": "Meta-Llama-3.1-8B-Instruct"
  },
  "system_prompt": "You are an expert in
{{domain_of_expertise}}. Please help the users
with their requests.",
  "tools": [{
    "component_type": "ServerTool",
    "name": "rag_tool",
    "description": "Tool that performs RAG",
    "inputs": [{"title": "query", …}],
    "outputs": [{"title": "results", …}]
  }],
  "agentspec_version": "25.4.1"
}
```

LangGraph Adapter

```
def convert_agentspec_llm_to_langgraph(agentspec_llm):
    from langchain_openai import ChatOpenAI
...
def convert_agentspec_agent_to_langgraph(agentspec_agent):
    from langgraph.prebuilt import create_react_agent
...
agent = convert_agentspec_agent_to_langgraph(agentspec_agent)
```

AutoGen Adapter

```
def convert_agentspec_llm_to_autogen(agentspec_ llm):
    from autogen_ext.models.openai import
OpenAIChatCompletionClient
...
    from autogen_agentchat.agents import AssistantAgent
...
agent = convert_agentspec_agent_to_autogen(agent_agent)
```

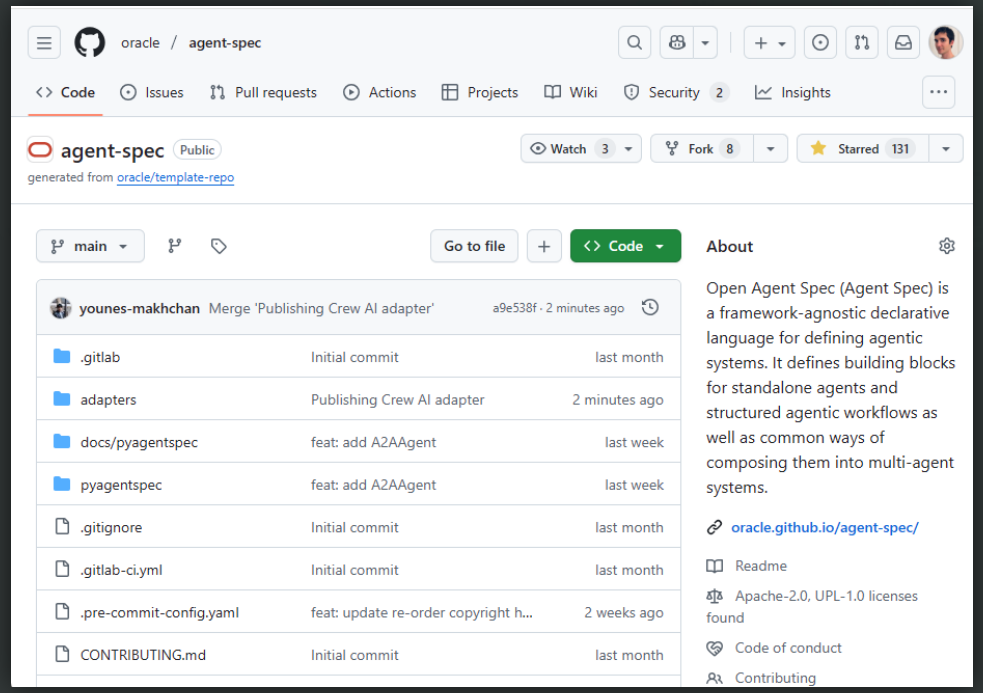
WayFlow Adapter

```
def convert_agentspec_llm_to_wayflow(agentspec_llm):
    from wayflowcore.models import VllmModel
...
...
agent = convert_agentspec_agent_to_wayflow(agentspec_agent)
```

Open Agent Specs – Github & PyPI

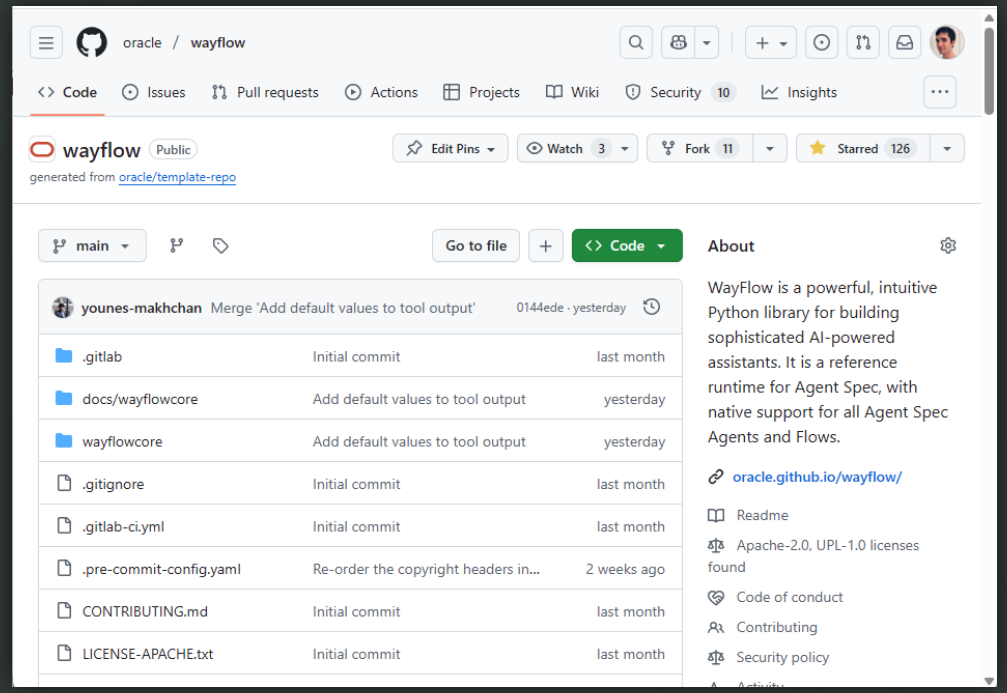
Source code and documentation are available on github; Installable directly from PyPI;

```
$ pip install pyagentspec
```



Open Agent Specification
https://github.com/oracle/agent-spec

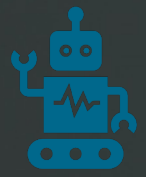
```
$ pip install wayflowcore
```



WayFlow (runtime for AgentSpec)
https://github.com/oracle/wayflow

Open Agent Specs – Code Example – Agents

Code example with pyagentspec



HR Agent

Tools

User

question →

← answer

The assistant is simply defined with the list of tools, the custom instructions as well as the LLM to use

```
HRAGENT_INSTRUCTIONS = """You are a
knowledgeable, factual, and helpful HR
assistant that can answer simple HR-
related questions like salary and
benefits."""

assistant = Agent(
    name="HR Assistant",
    system_prompt=HRAGENT_INSTRUCTIONS,
    tools=[search],
    llm_config=llm,
)
```

tool calls ⇠ - - - ⇢

Assistant is equipped with tools to enhance their capabilities

```
search = ServerTool(
    name="search_tool",
    description=(
        "Retrieve information about "
        "an employee."
    ),
    inputs=[
        StringProperty(name="username")
    ],
    outputs=[
        StringProperty(name="user_details")
    ],
)
```

Open Agent Spec – Components – Tools

Supported tool types

Agent Spec makes it easy to integrate into existing software stacks with several types of tools

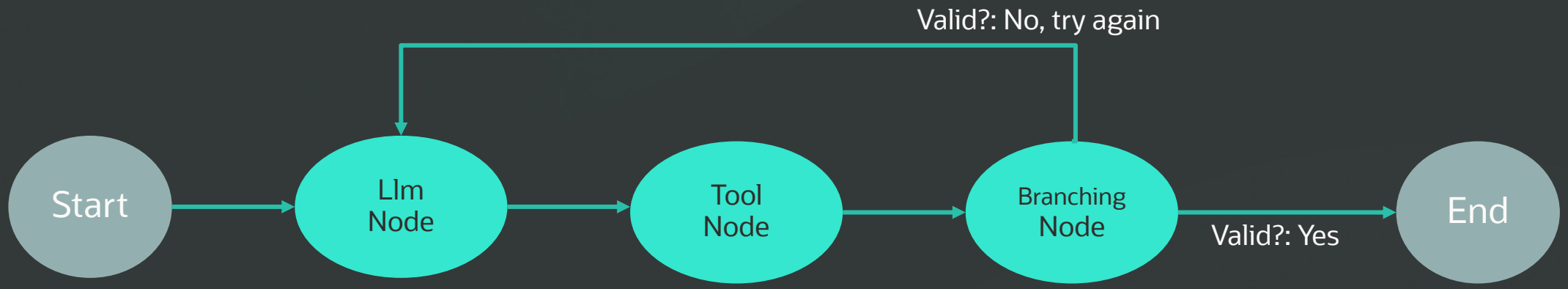
Tool Type	Description	Agent Spec Tool Configuration
MCP	Interacts with **Model Context Protocol (MCP)** servers for remote tool execution.	Tool metadata (name, description, inputs, outputs) + URL and communication protocol for the MCP server hosting the tool
Remote	Tool logic runs in an **external environment**, triggered via RPC or REST.	Tool metadata (name, description, inputs, outputs) + Details of the API request to be performed (e.g. url, payload, ...)
Client	Tool is **run by the client**, which returns results to the executor.	Tool metadata (name, description, inputs, outputs)
Server	Tool runs in the **same runtime** as the agent.	Tool metadata (name, description, inputs, outputs) + Requires tool implementation for execution

Open Agent Specs – Components – Flows

Flows specify the steps of a workflow and transitions between them

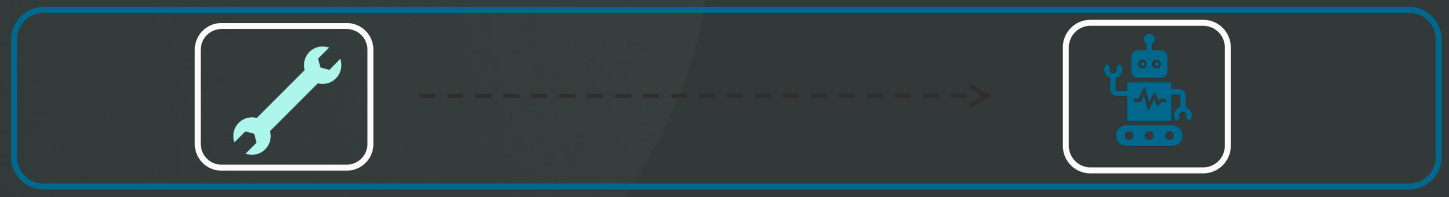
Control and data flow are constrained by dedicated edge types and define respectively the order of execution of nodes and how data is passed from outputs to inputs of the following nodes

Some nodes might use LLMs, some nodes not (e.g. do REST API call).



Open Agent Specs – Code Example – Flow

Code example with pyagentspec



A Flow is defined with nodes and transitions between the nodes

```
start_node = StartNode(name="start", inputs=[query_property])
tool_node = ToolNode(
  name="hr_lookup_node",
  tool=search_hr_database,
  outputs=[results_property]
)
llm_node = LlmNode(
  name="llm_answer_step",
  llm_config=llm_config,
  prompt_template="Answer the user given the question {{
query }} with the given context: {{ hr_context }}",
  inputs=[query_property, hr_context_property],
  outputs=[answer_property],
)
end_node = EndNode(name="end", outputs=[answer_property])
```

```
control_flow_edges = [
    ControlFlowEdge(name="tool_to_llm",
from_node=tool_node, to_node=llm_node), …
]

data_edges = [
    DataFlowEdge(name="query_data_edge",
source_node=start_node, source_output="query",
destination_node=tool_node, destination_input="query"),  …
]

flow = Flow(
    name="HR Flow",
    start_node=start_node,
    nodes=[start_node, tool_node, llm_node, end_node],
    control_flow_connections=control_flow_edges,
    data_flow_connections=data_flow_edges,
)
```

Serialization

Pre-built components have a serialized representation (e.g. JSON)

Flow configurations:

- List of nodes
- Control/data flow definition

Node configurations:

- Inputs/outputs definition
- Per node configuration

LLM configurations:

- Host type, URL
- Model Type
- Generation parameters

Flow information

Control flow definition

Data flow definition

Nodes in flow

```
{ "component_type": "Flow",
  "inputs": [ {"title": "article", "type": "string"}],
  "outputs": [{"title": "summary", "type": "string"}, ...],
  "nodes": [
    {"$component_ref": "start_node_id"},
    {"$component_ref": "llm_node_id"},
    {"$component_ref": "end_node_id"}],
  "control_flow_connections": [
    {
      "component_type": "ControlFlowEdge",
      "from_node": {"$component_ref": "start_node_id"},
      "to_node": {"$component_ref": "llm_node_id"}
    }, ...],
  "data_flow_connections": [
    {
      "component_type": "DataFlowEdge",
      "source_node": {"$component_ref": "start_node_id"},
      "source_output": "article",
      "destination_node": {"$component_ref": "llm_node_id"},
      "destination_input": "article"
    }, ...],
  "$referenced_components": {
    "llm_node_id": {
      "component_type": "LlmNode",
      "inputs": [{"title": "article", "type": "string"}],
      "outputs": [{"title": "summary", "type": "string"}, ...],
      "llm_config": {
        "component_type": "OllamaConfig",
        "default_generation_parameters": {"max_tokens": 512, "temperature": 0.7},
        "url": "http://localhost:11434",
        "model_id": "llama3.1"
      },
      "prompt_template": "Extract the relevant information from the article: {{article}}"}, ...},
  "agentspec_version": "25.4.2"}
```

Open Agent Specs – Composability and Multi-Agent Systems

From single agents to reusable multi-agent systems

Agent Spec enables developers to **reuse modular components** when building new assistants.

Composability Patterns:

- Flow-in-Flow enables to develop complex logic with well separated reusable subflows
- Agent-in-flow integrates agents into orchestrated workflows
- Flow-in-agent lets agents invoke complex workflow in the same manner tools are invoked
- Swarm / Manager–Worker supports intelligent query routing and specialization

Upcoming support for the **A2A protocol** standardizes communication across agents

Open Agent Specs – Built-in Components

Agent Spec supports a large number of built-in components, and more are being added in future versions

Flow & Node Types

- `Flow`
- `AgentNode`
- `ApiNode`
- `BranchingNode`
- `EndNode`
- `FlowNode`
- `InputMessageNode`
- `LlmNode`
- `MapNode`
- `OutputMessageNode`
- `StartNode`
- `ToolNode`
- `...`

Other component Types

- `Agent`
- `OpenAiConfig`
- `VllmConfig`
- `OllamaConfig`
- `OciGenAiConfig`
- `ClientTool`
- `ServerTool`
- `RemoteTool`
- `MCPTool`
- `Property`
- `Swarm`
- `...`

Open Agent Specs – Evaluation Harness

Evaluation Harness - https://arxiv.org/pdf/2510.04173v3

Agent Spec enables comparison of agent and flow patterns across different frameworks.

We ran this evaluation harness on representative tasks from three benchmark datasets:

- LangGraph and WayFlow deliver reliable accuracy and query time stability across these datasets
- AutoGen outperforms on complex reasoning tasks (e.g., on τ2 - Bench) but with increased latency
- CrewAI's multi-agent coordination yields limited accuracy gains despite longer query times.

Agent Setup	Runtime	F1-score (%)	Time/query (s)
GPT-4.1 mini	-	17.8	1.06 ± 0.41
ReAct Agent (GPT-4.1 mini)	WayFlow	79.0	6.22 ± 5.24
	LangGraph	75.5	4.57 ± 6.21
	AutoGen	67.9	4.64 ± 1.64
	CrewAI	**81.1**	7.21 ± 6.74
ReAct Agent (Llama 3.3 70B)	WayFlow	79.2	4.60 ± 0.92
	LangGraph	77.6	4.10 ± 0.91
	AutoGen	54.9	3.38 ± 0.31
	CrewAI	**82.3**	8.12 ± 3.84
Flow-based agentic RAG (GPT-4.1 mini)	WayFlow	**88.1**	23.19 ± 43.25
	LangGraph	85.3	23.21 ± 55.53

Benchmark results on SimpleQA Verified

Agent Setup	Runtime	EX (%)	Time/query (s)
GPT-4.1 mini	-	50.1	4.26 ± 2.25
GPT-4.1	-	54.4	2.93 ± 4.68
ReAct Agent (GPT-4.1)	WayFlow	54.1	7.16 ± 2.72
	LangGraph	53.7	5.68 ± 3.64
	AutoGen	54.1	6.06 ± 2.97
	CrewAI	**54.3**	12.20 ± 6.62
NL2SQL Flow (GPT-4.1)	WayFlow	55.5	9.45 ± 5.56
	LangGraph	**55.9**	7.54 ± 4.11

Benchmark results on BIRD-SQL

Agent Setup	Runtime	Pass^1 / ^4 (%)	Time/query (s)
ReAct Agent (GPT-4.1)	WayFlow	64.7 / 43.0	43.0 ± 13.8
	LangGraph	71.5 / 51.8	37.5 ± 12.4
	AutoGen	**73.5** / **52.6**	55.5 ± 17.8
	CrewAI	62.5 / 38.6	62.7 ± 25.4
ReAct Agent (GPT-4.1 mini)	WayFlow	60.1 / 32.5	40.9 ± 12.7
	LangGraph	**60.7** / 32.5	38.4 ± 17.4
	AutoGen	58.3 / **36.0**	47.4 ± 15.0
	CrewAI	51.5 / 22.8	56.1 ± 25.5

Benchmark results on τ2 -Bench

Open Agent Specification – Technical Report
https://arxiv.org/pdf/2510.04173v3

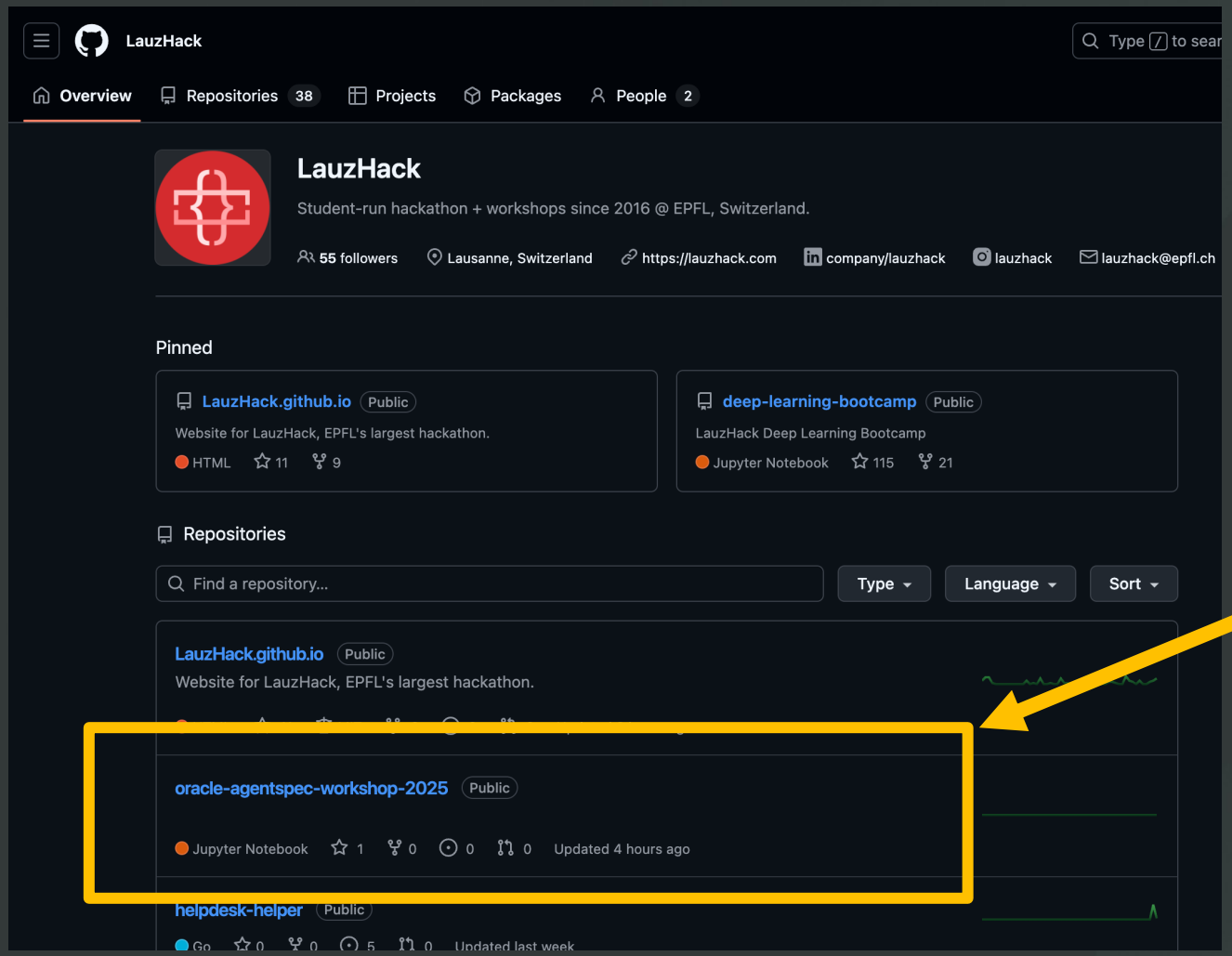
Open Agent Specs – Conclusion

Why trying out Agent Spec?

- Framework agnostic
- Sharable agents and flows specification in JSON format
- It's open Source
- Many built-in components that work out of the box
- Support for most types of LLMs models
- Extensive documentations and how-to guides
- Allows you to develop SOTA Agents and Flows

Demos

Demos – Repository



You can clone the repository on LauzHack's GitHub to execute the code yourself or just follow along with me.

Conclusion

Conclusion

Want to learn more about AI Agents ?

- Read our medium blog posts
- Check the docs and guides: https://oracle.github.io/agent-spec/

Want to contribute to an Open Source agentdevelopment framework?

- AgentSpec: https://github.com/oracle/agent-spec
- Wayflow: https://github.com/oracle/wayflow

Want to get more practice and hands-on with AgentSpec & Wayflow?

- If you have registered to the LauzHack Hackathon, come solve Oracle's challenges!

Interested in Internships & Full-time positions at Oracle?

- Send your interests to me: louis.faucon@oracle.com


ORACLE