⚡ Lightning Agents ⚡

Factory-of-Factories for Dynamic AI Agents

Ricardo Pirruccio

github.com/RPirruccio/lightning-agents | aimug.org

The Problem

TYPICAL AGENT CODE

✗ agent_v1.py, agent_v2.py, agent_final.py...

✗ Hardcoded prompts in every file

✗ Copy-paste to create new agents

✗ Change model name in 47 places

✗ No single source of truth

FACTORY-OF-FACTORIES

✓ One agents.json file

✓ Declarative definitions

✓ Registry builds factories

✓ Runtime context injection

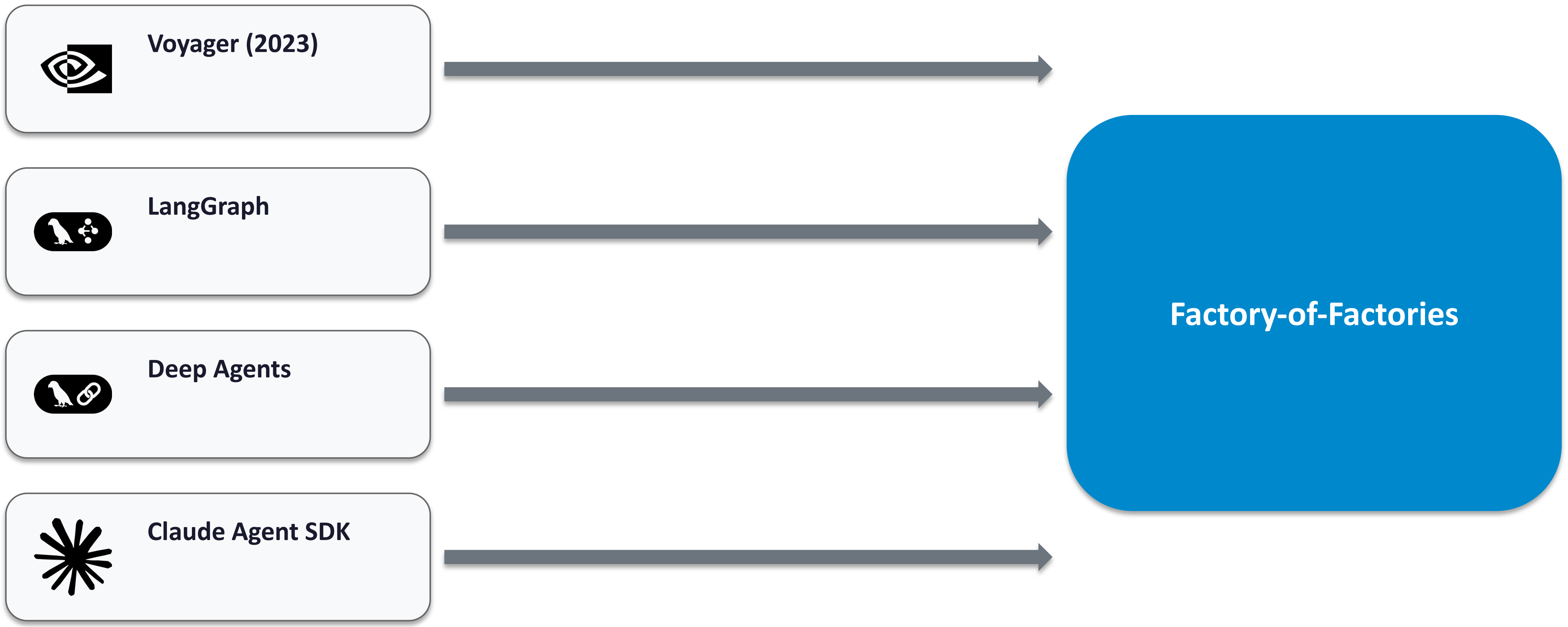
✓ Agents create new agents

The Voyager Insight

VOYAGER (2023)	LIGHTNING AGENTS
Learns new skill	Creates new agent
Stores in skill library	Stores in agents.json
Retrieves when needed	Registry.create()
Skills compound over time	Agents create agents

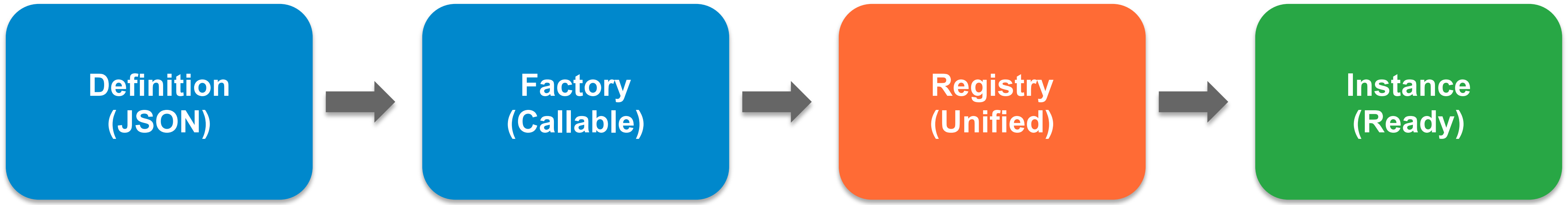
SAME PATTERN. DIFFERENT DOMAIN.

The Industry Agrees



Everyone's arriving at the same place: less code, more context engineering

The Solution: Factory-of-Factories



Simple Factory vs Factory-of-Factories

Simple Factory

```
def create_agent(config):
    return Agent(config)
```

Factory-of-Factories

```
registry = AgentRegistry.from_json(
    "agents.json"
)
agent = registry.create(
    "researcher", {"topic": "AI"}
)
```

Declarative Agent Definition

```json
{
  "architect": {
    "name": "Agent Architect",
    "description": "Designs new agents",
    "system_prompt": "You are an Agent Architect...",
    "model": "sonnet",
    "tools": [
      "mcp__custom-tools__db_create_agent",
      "mcp__custom-tools__db_list_agents",
      "mcp__custom-tools__run_agent"
    ]
  }
}
```

The Registry Pattern

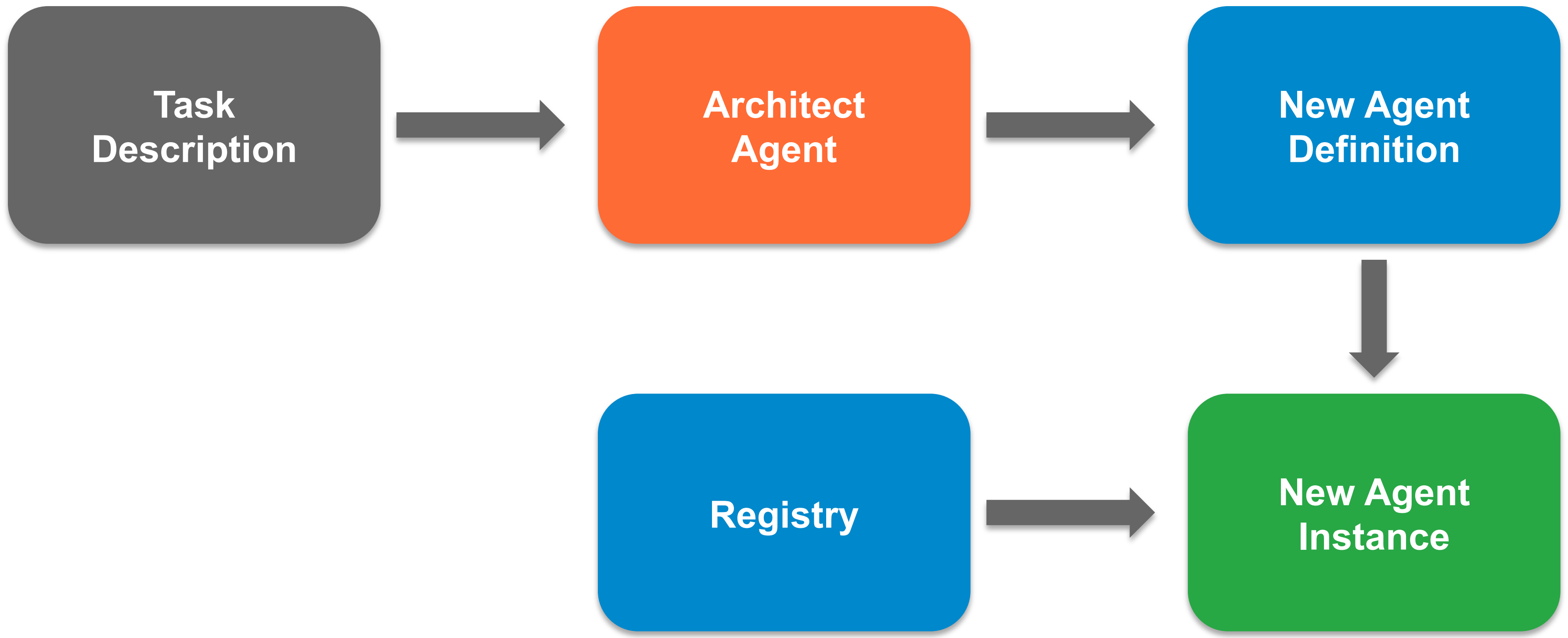
```
class AgentRegistry:
    def from_json(path) -> "AgentRegistry":
        # Load definitions -> build factories

    def create(id, opts) -> AgentInstance:
        # Factory creates configured instance

    def register(id, defn) -> None:
        # Add new agent to registry
```

Single source of truth for all agent definitions

Runtime creation - agents spawn new agents on demand

Self-modifying - architects can register new agents

The Architect Agent



Introducing: The Tool Architect

Agents can also design **NEW TOOLS**

`tool_architect` registers definitions in `db/tools.json`

`paper_researcher` searches papers **AND** downloads PDFs

`presentation_slide_writer` manipulates slides **directly**

Demo: What We Built

```
$ lightning list
  10 agents: architect, tool_architect, paper_researcher...

$ lightning run paper_researcher "Find the Voyager paper"
  Found: arxiv.org/abs/2305.16291
  Downloaded: voyager_lifelong_learning.pdf

$ lightning run presentation_slide_writer "List slides"
  12 slides in current presentation...
```

The Meta Moment

We used `paper_researcher` to find the **Voyager paper**

Voyager inspired the **architect pattern**

We built `tool_architect` to create **more tools**

`presentation_slide_writer` built **THIS presentation**

This slide was created by an agent that was created by an agent

⚡ Lightning Agents ⚡

Questions?

Agents creating **agents** creating **tools**

Built with `Claude Agent SDK` + `MCP`

`github.com/aimug-org/austin_langchain`