

EVOLUTION OF AGENT DESIGN

The Importance of Agent Design in Copilot Studio

Building agents with subagents



S P E A K E R

Introduction



Cathrine Bruvold

Head of Power Platform 🚀 **Point Taken**
Business Applications **MVP**



Norway



github.com/cathrinebruvold



linkedin.com/in/bruvold

S P E A K E R

Introduction



Andrew Bibby

Consulting Director, **Proximo 3**
Business Applications **MVP**



United Kingdom



[linkedin.com/in/andrewbibby](https://www.linkedin.com/in/andrewbibby)

What this session will cover

01 The evolution of agents

02 What is a multi-agent system?

03 Subagents, connected agents, or both?

04 Live demo: building a governance agent

05 The importance of instructions

06 What to remember about agent design

The evolution of agents

FROM RULES

Previously the focus was on rule-based agents, one **monolith** agent with all the instructions, tools and knowledge sources. A “**chatbot**”.

...TO REASONING

Shift in the design approach to many **specialist agents** working together, avoiding over-complicating one task-oriented agent. Moving to **generative agents** performing tasks.

THE CHANGE

Architecture and design recommendations has changed and deciding on how to approach building an agent is more important now than before.

What is a “multi-agent system”?

A NETWORK OF AGENTS COLLABORATING

The main agent, the **Orchestrator**, is responsible for dividing tasks by **delegating responsibility** to other agents.

Avoiding overloading **one** agent with too much complexity and simplifying the design to ease the process of letting the LLM handoff a request to another agent.

AGENT-TO-AGENT

Copilot Studio supports **multi-agent systems** for collaboration between agents, allowing Makers to design for scale, isolate responsibility and ease the process of **orchestration**.

The “Russian doll” pattern

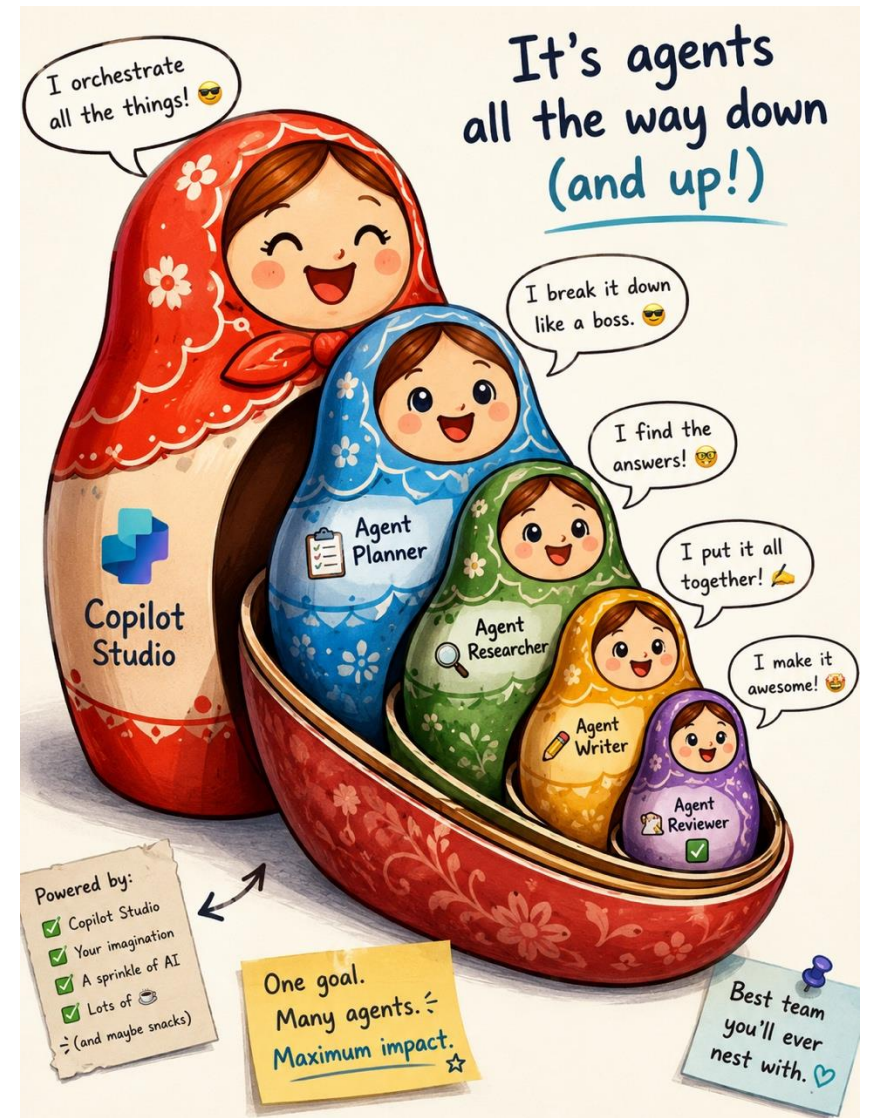
HIERARCHICAL PARENT-CHILD ARCHITECTURE

Subagents, Inline agents, child agents, specialist agents, embedded agents...

Each agent is built *within* the **Orchestrator agent**, and has a clear role and scope – it becomes a specialist, not a generalist

The subagents inherit the settings of the Orchestrator agent and will only trigger if its description matches the user intent.

A specialist agent can also hand off to another agent and route requests as part of a chain.

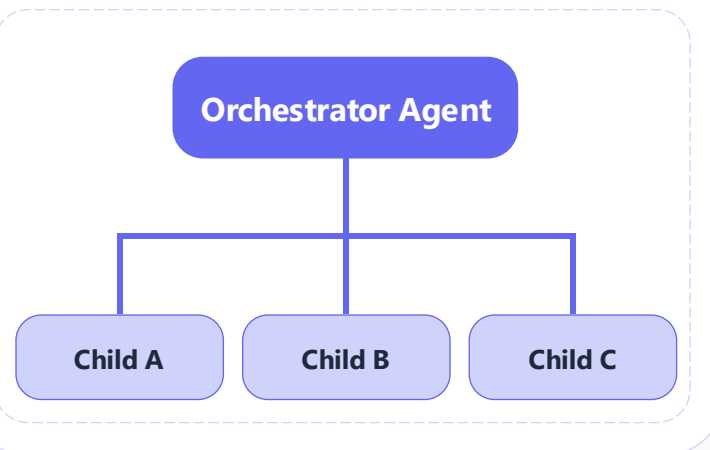


Architecture: subagent, connected, or both?

APPROACH 01

Subagents

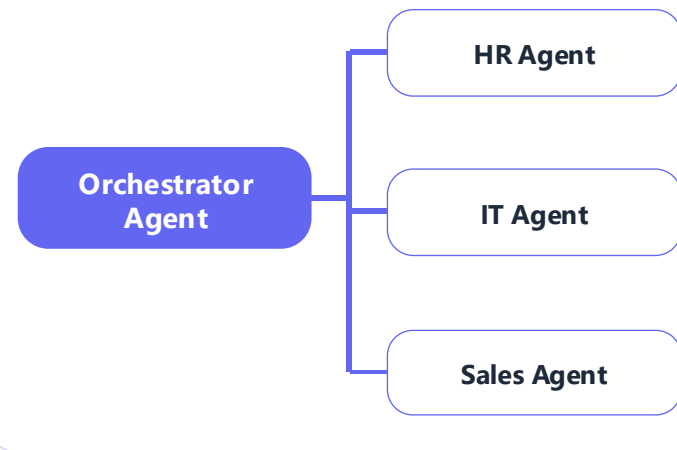
Child agents created inside the orchestrator.



APPROACH 02

Connected agents

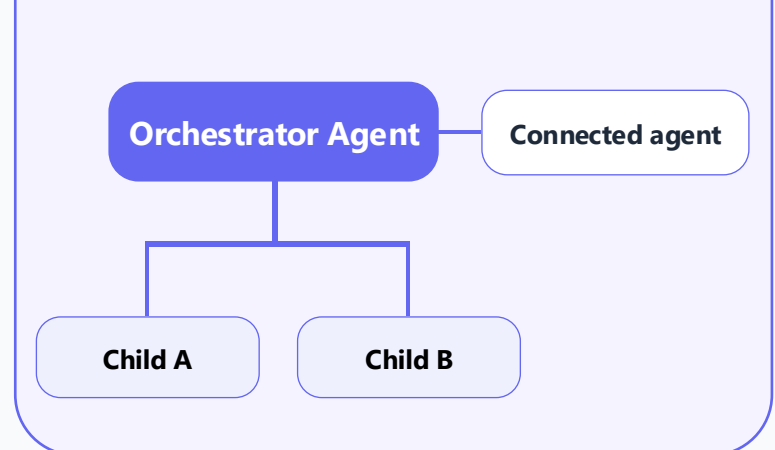
Independently published agents called by the orchestrator.



APPROACH 03 - HYBRID

Combined both

Inline children for tight, custom logic. Connected agents for shared, reusable capabilities.



Which should I use?

Subagents

Break into focused child agents when:

- One developer or small team manages the whole solution
- You want to logically group tools, instructions or knowledge into clearly defined child agents
- Need for custom agents that possibly could change over time
- Light weight, less set up.

Connected agents

Standalone

Use connected agents when:

- Multiple teams manage different agents independently
- Agents need to be published and available on their own channels
- Each agent requires its own dedicated settings or authentication configuration
- Independent ALM (Application Lifecycle Management) per agent

⚠ **Shared agent = shared changes** Updating a connected agent impacts everyone using it

One long block of instructions vs delegation

One monolith agent – will perform every task

Instructions Cancel Save

↶ ↷ + Add

4. Document Actions

- Log all findings and recommendations in a governance report.
- Share the report with the admin team for review.

Error Handling

- If unable to retrieve data, check API permissions and connectivity.
- If user status cannot be verified, flag the resource for manual review.

Interaction Examples

- **Admin:** "Find orphaned apps in the environment."
- **Agent:** "I have identified 3 apps without active owners. Would you like to reassign or archive them?"

Nonstandard Terms

- **Orphaned Resource:** A resource without an active owner.
- **CoE Starter Kit:** A Microsoft toolkit for Power Platform governance.

Follow-up and Closing

- Always confirm before taking any action on resources.

2069/8000


Orchestrator delegating tasks to child agents


Instructions Cancel Save


↶ ↷ + Add

You're a governance agent with specialist agents.


##Child agents

Discovery Agent
Hand off to the  Discovery Agent when the user wants to scan for orphaned resources. This includes requests to find, search, detect, check, or identify orphaned flows, apps, or agents across one or all environments.

Inventory Agent
Hand off to the  Inventory Agent when the user wants to see or browse what has already been found. This includes requests to show, list, display, view, or filter orphaned resources by type or environment.

Remediation Agent
Hand off to the  Remediation Agent when the user wants to delete or remove a specific orphaned resource. Only route here when the user has identified a specific resource to act on. This includes requests to delete, remove, or clean up a named resource

Rules

If the user asks to see results before any scan has been run, hand over to the  Inventory Agent, if no results tell them no results exist yet and

1059/8000

Governance agent with specialist subagents

USE CASE

Overview of orphaned items in Power Platform environments. Orphaned assets are flows, agents and apps whose owner is a disabled user, and that need to be discovered, inventoried and safely deleted.

ORCHESTRATOR • PARENT

The Governor

Routes user messages to the right specialist.

SUBAGENT 01 • RESEARCH

Discovery Agent

Analyses environments for orphaned items. Looks for elements where the owner is a disabled user.

SUBAGENT 02 • INVENTORY

Inventory Agent

Tracks everything marked as orphaned and collects the details required to decide on next steps.

SUBAGENT 03 • DELETION

Remediation Agent

Once discovery and inventory are complete, this agent is tasked with deleting the item.

LIVE DEMO

Building a Governance Agent and Writing Instructions

Hand off to specialist agents

The Importance of Instructions

Name your tools and agents in the orchestrator instructions

Example: "Use the Discovery agent when the user asks about scanning environment or orphaned items"

Use / to reference agents and tools directly in instructions

Type / inside the instructions field to get a picklist of all available tools and connected agents

Good descriptions drive smart routing

The LLM uses each tool/agent description to decide when to call it. Write **descriptions** like decision rules


Test routing in the Activity Map

After configuring, use the Copilot Studio test pane Activity Map to confirm the right agent or tool is being invoked

Instructions



You are the Discovery Agent. Your only job is to scan Power Platform environments for orphaned r

When invoked, ask the user if they want to scan all environments or a specific one. If they say all, p
want a specific environment, ask them to name it and then trigger the  Governance - Scan Orphan
received a environment ID, it looks like something like this: Default-681e41cd-3aea-474d-9e84-920

When the scan completes, report back how many orphaned resources were found, broken down b
and how many environments were scanned based on the output of the Scan Flow. Tell the user tha
inventory to proceed.

Do not show the full list of resources — that is the Inventory Agent's responsibility. Only report the

In the solution

Subagent visibility

*A child agent is shown as a **Topic***

...meaning that these agents (and Agent Flows) are not showing as what they truly are as they live within the scope of the parent agent, the orchestrator

Power Platform Governance > All

Display name ↑	Name	Type	Managed
Conversation Start	Conversation Start	Topic	No
Conversational boosting	Conversational b...	Topic	No
Discovery Agent	Discovery Agent	Topic	No
End of Conversation	End of Conversat...	Topic	No
Escalate	Escalate	Topic	No
Fallback	Fallback	Topic	No
Goodbye	Goodbye	Topic	No
Governance - Delete App	Governance - De...	Topic	No
Governance - Get Inventory	Governance - Ge...	Topic	No
Governance - Scan Orphaned Resources	Governance - Sc...	Topic	No
Greeting	Greeting	Topic	No
Inventory Agent	Inventory Agent	Topic	No
Multiple Topics Matched	Multiple Topics ...	Topic	No
On Error	On Error	Topic	No
Remediation Agent	Remediation Age...	Topic	No
Reset Conversation	Reset Conversati...	Topic	No
Sign in	Sign in	Topic	No

SUMMARY

What to remember about agent design

01 PRINCIPLE

Specialists over generalists

One agent doing everything becomes impossible to debug. Scope each agent to a clear job - it's easier to test, troubleshoot, and improve

02 ARCHITECTURE

Inline, connected, or both

Inline children for tight, custom logic. Connected agents for shared, reusable capabilities. Don't split things just because you can

03 INSTRUCTIONS

Write. Good. Instructions.

The orchestrator gets a simple brief; specialists get scoped ones. Agent and tool descriptions are how the orchestrator knows where to send work

04 OPERATIONS

Governance and ownership

Who owns each agent, who can change it, and what happens when the owner leaves? Build for the entire lifecycle, not just for flashy demos

05 HANDOFFS

Design for handoffs

Agents collaborate; they don't compete. Build clean handoff points so work flows between specialists without dropping context.

06 DESIGN

Architecture over prompts

Scale responsibilities, not instructions. When prompts get long, that's a signal to split the agent – not to add more instructions

Resources & links

01

Multi-agent orchestration patterns

Serial, concurrent, agentic & A2A – official guidance

learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/multi-agent-patterns

02

Orchestrator-subagent pattern

The Russian doll / hierarchical pattern – deep-dive

learn.microsoft.com/microsoft-copilot-studio/guidance/architecture/multi-agent-orchestrator-sub-agent

03

Agent Academy

Hands-on curriculum to learn building agents in Copilot Studio

aka.ms/agent-academy

04

Generative orchestration guide

LLM-driven planning, instructions & testing your agent

learn.microsoft.com/en-us/microsoft-copilot-studio/guidance/generative-mode-guidance



Q&A

**Thank you for
attending!**