From Multi-Agent Patterns to Reliable Orchestration

Orchestration depends on one repeated decision: who should handle this piece of work?

Daniel Homola · Lead AI Engineer, BMW Research

My Recent Work on Agents at BMW

RECENT FOCUS

Agentic AI architecture

GUI agents that operate screens

RELATED TALK

LLM-Based GUI Agents

AI Engineer Paris 2025

A Multi-Agent In-Car Assistant

Hypothetical example: what agents could naturally exist in any modern vehicle?

🗺️ **Navigation Agent**

Routing & destinations

🚗 **Car Control Agent**

Windows, climate, seats

🖥️ **GUI / Computer-Use Agent**

Operates apps via screen

The orchestrator chooses who acts, coordinates execution, and may combine results.

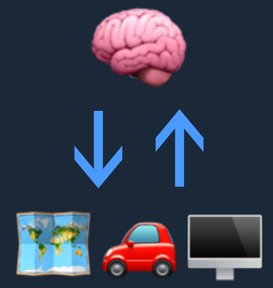
The Patterns: How Agents Coordinate

Patterns can be mixed. The challenge is the runtime decision.

HANDOFF

Control transfers.

AGENTS AS TOOLS



Delegate work, await result.

ROUTING

Dispatch each turn.

Swarm / Consensus · Pipeline · Evaluator–Optimizer · Broadcast · *...and more*

Runtime decision:

which agent or tool should handle this piece of work?

Delegation Is More Than Tool Selection

Same selection interface, different decision.

Tool Selection

"Which tool **fits**?"

Tools should not overlap

One label should work

Agent Delegation

"Who should handle **this work**?"

Overlap is normal

Cost, speed, reliability & UX matter

User request: *"Play some jazz"*

Empty screen → Media API. Jazz playlist visible → GUI tap can also be valid.

Overlap + context create alternatives.

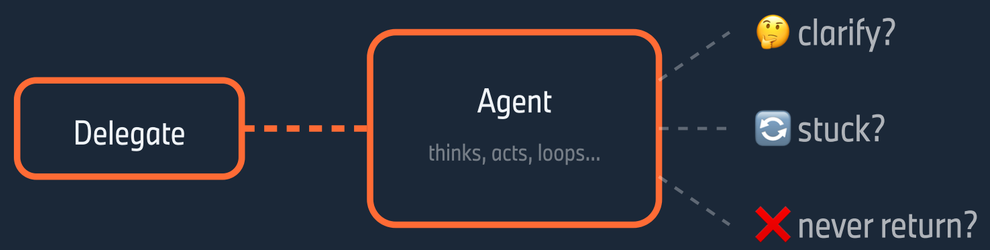
Agents are not tools

Even when delegation is *implemented* as a tool call, it isn't one.

Tool Call

One step. Control returns to the caller.

Agent Delegation



Control leaves. May not return.

Bounded function execution. The caller passes input, gets one result, and keeps control.

Control leaves to an autonomous loop that may ask, act, resume, or never return.

Same tool-calling interface. Different control semantics.

Where the Decision Gets Ambiguous

The GUI agent is a useful case study: it can reach outcomes through the same UI the user sees.

User Request	GUI Path	Specialist Path	Best?
"Navigate to Munich"	Maps UI → go	Navigation API	Specialist ✓
"Close the windows"	Settings UI	Vehicle API	Specialist ✓
"Open that restaurant's site"	Use browser screen	No specialist for this	GUI Agent ✓
"Play some jazz" (music app open)	Tap visible playlist	Media API	Ambiguous ⚖️

The point: overlapping agents can make several paths valid; context helps choose.

A Possible Way to Evaluate Orchestration & Delegation

A direction, not a recipe.

The dataset
User request + full runtime context (history, app/vehicle state, available agents & permissions) → acceptable agent/tool choices.

The metric
Accept valid choices, but evaluate cost, speed, reliability, and UX.

Enterprise path: each subteam builds evals for the agent it owns.
Together with the orchestrator team, they add cross-domain collaboration cases; the union becomes one shared orchestration & delegation benchmark.

Reliable orchestration depends on reliable delegation decisions.

Who should handle this piece of work?

Daniel Homola · BMW Research

LET'S CONNECT