PROGRAMME BOOK SERIES · No. 268

Perspective

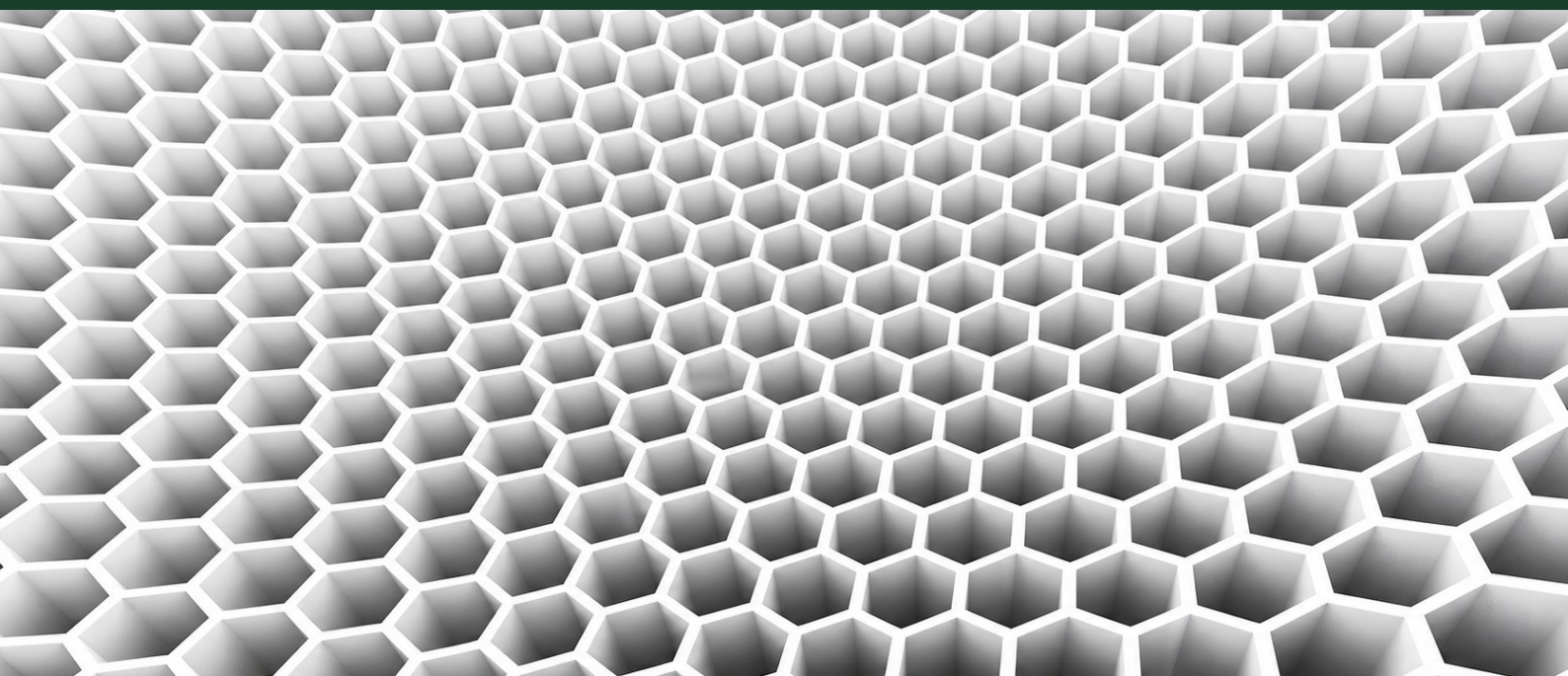
BEYOND AGENT COORDINATION

Multi-Agent Orchestration Fails When Nobody Owns the Hand-Offs

Why enterprise agent systems need decision contracts, not just specialised roles

Giovanni Leonardi

June 2024



Multi-Agent Orchestration Fails When Nobody Owns the Hand-Offs

Programme · No. 268
Originally written June 2024 | Re-edited and re-rendered for the WhereBeyond Edition,

"The failure point in a multi-agent system is usually not the reasoning inside an agent but the meaning lost between agents."

The hand-off nobody was watching

A delivery team connects five specialised agents to prepare a weekly change assessment. One retrieves requirements, one examines dependencies, one drafts risks, one challenges the draft, and one assembles the final recommendation. In a demonstration, the chain looks disciplined. Every agent has a role; every output arrives in seconds.

In the third week, a requirement marked "pending commercial approval" becomes "approved" by the time it reaches the risk agent. No agent invented the approval. The retrieval agent shortened a status field, the dependency agent treated the shortened phrase as current state, and the final agent inherited the assumption. The recommendation is fluent, internally consistent and wrong.

That small sequence captures what the early enterprise experiments of 2024 are teaching us. Multi-agent orchestration is presented as a problem of assigning roles, selecting tools and controlling sequence. In practice, it is a problem of preserving meaning, authority and accountability across hand-offs.

The failure point in a multi-agent system is usually not the reasoning inside an agent but the meaning lost between agents.

Specialisation solves the visible problem

The case for multiple agents is serious. A single general-purpose prompt becomes difficult to test as the task grows. Specialised agents can separate retrieval from analysis, analysis from challenge, and recommendation from execution. They offer modularity: a team can improve one role without rewriting the entire workflow. They can also create useful tension by asking one agent to critique another.

These are real advantages. The mistake is to assume that decomposition itself creates control.

Textbook diagrams tend to show clean arrows between boxes. Enterprise work is not clean. A project status can be provisional, disputed, stale or authorised only for one audience. A risk may be tolerable to a workstream and unacceptable to a programme. A statement may be evidence, interpretation or instruction. When an agent passes text without preserving those distinctions, orchestration accelerates ambiguity.

Human teams have the same problem, but they compensate through context. A programme manager hears hesitation in a meeting, remembers who owns a decision, and knows that "agreed" sometimes means "agreed in principle." Agents receive whatever state the workflow represents. If that state has flattened uncertainty, they inherit certainty they did not earn.

The four contracts that matter

The useful unit of design is not the agent role. It is the contract at the hand-off. Four contracts determine whether a multi-agent workflow remains governable.

Contract	Question	Minimum evidence
Meaning	What exactly is being passed?	Structured field, source, status and confidence
Authority	What may the receiver infer or do?	Permitted transformation and prohibited action
State	Which version of the world applies?	Timestamp, workflow version and unresolved conditions
Accountability	Who owns the consequence?	Named human decision owner and escalation route

A role description such as "risk agent" is therefore too weak. It says what the agent is for, but not what evidence it may treat as authoritative, which assumptions it must preserve, or when it must stop.

Consider a composite release-planning workflow handling 86 change items. The retrieval agent finds 79 complete records and seven with missing dependency approvals. A loosely designed hand-off sends one narrative summary downstream. The planning agent converts all 86 into a proposed sequence because its instruction is to optimise the plan.

A contracted hand-off behaves differently:

- approval state remains a separate field rather than being buried in prose;
- the source record and retrieval time travel with each item;
- "unknown" cannot be converted into "no dependency";
- the planning agent may sequence incomplete items only as conditional;
- the seven exceptions route to the dependency owner before any commitment.

The difference is not a better model. It is a better operating contract.

Orchestration is programme design

Teams often place multi-agent systems inside a technical architecture discussion. That is necessary but insufficient. Once agents contribute to prioritisation, assurance, scheduling, financial assessment or stakeholder communication, the workflow has entered programme governance.

Three consequences follow.

A conventional process map shows activities. An agentic workflow also needs a decision topology: where judgement occurs, where evidence changes form, where authority increases, and where an automated action becomes externally consequential.

The programme manager should be able to point to every place where:

- a fact becomes an inference;
- an inference becomes a recommendation;
- a recommendation becomes an action;
- an exception can stop the chain;
- a human assumes accountability.

If those transitions exist only inside prompts and routing code, the programme has delegated its governance model to implementation detail.

Logging each agent's final message is not enough. The programme needs to reconstruct the path: input, source, intermediate state, transformation, tool call, exception and final action. Otherwise incident review becomes a collection of plausible explanations.

This does not require retaining every internal token. It requires retaining the organisational evidence needed to answer: what was known, what changed, what authority applied and why the workflow proceeded.

Demonstrations are built around the happy path. Enterprise delivery is shaped by partial information, contested ownership and policy conflict. The quality of orchestration is revealed when two agents disagree, a source is missing, confidence is low, or the next action is irreversible.

> *The route for disagreement and uncertainty is not a fallback around the system; it is the part that makes the system safe enough to use.*

An exception should not automatically summon another agent until consensus appears. Agreement among agents can be repeated error. Some exceptions require a human decision owner, a request for new evidence, or termination of the workflow.

What the textbooks leave out

The cleanest multi-agent architecture can still fail because enterprise delivery contains politics, incentives and unequal authority. An agent asked to "challenge" a plan does not possess the standing of an assurance lead. An agent asked to "approve" a change does not carry delegated financial authority. Naming the role does not confer the institution.

This is where the language of autonomous teams can mislead. Agents may perform bounded cognitive work with considerable independence. The organisation remains responsible for defining which outputs have standing. A generated risk opinion can inform a decision; it does not become governance merely because a workflow labels it "reviewed."

The strongest counterargument is that too much contractual detail will destroy the speed and adaptability that make agents attractive. It can. A hand-off schema with dozens of mandatory fields can become a brittle form of process automation. The answer is not maximum structure. It is minimum sufficient structure at points of consequence.

Preserve provenance when claims move between agents. Preserve uncertainty when interpretation is incomplete. Require explicit authority before actions change systems, money, commitments or communications. Allow freer exploration where outputs remain reversible and advisory.

The programme leader's test

Before approving a multi-agent workflow, ask for one case to be replayed from source to consequence. Then ask for an incomplete case, a disputed case and a case where the

agents disagree. Do not judge only whether the final answer sounds sensible.

The test is whether the workflow can show:

- what each agent received;
- what it was permitted to change;
- which uncertainty remained;
- why the next step was allowed;
- who owns the resulting decision;
- how the chain stops and recovers.

If the team cannot answer those questions, it has orchestrated activity, not accountability.

Multi-agent systems may become a powerful way to divide complex work. But the enterprise advantage will not come from adding more specialised reasoners to the chain. It will come from designing the boundaries between them with the same seriousness we apply to decision rights, controls and programme dependencies. In complex delivery, the hand-off has always been where intent decays. Agents make that old truth faster, quieter and more consequential.