PROGRAMME BOOK SERIES · No. 267

White Paper

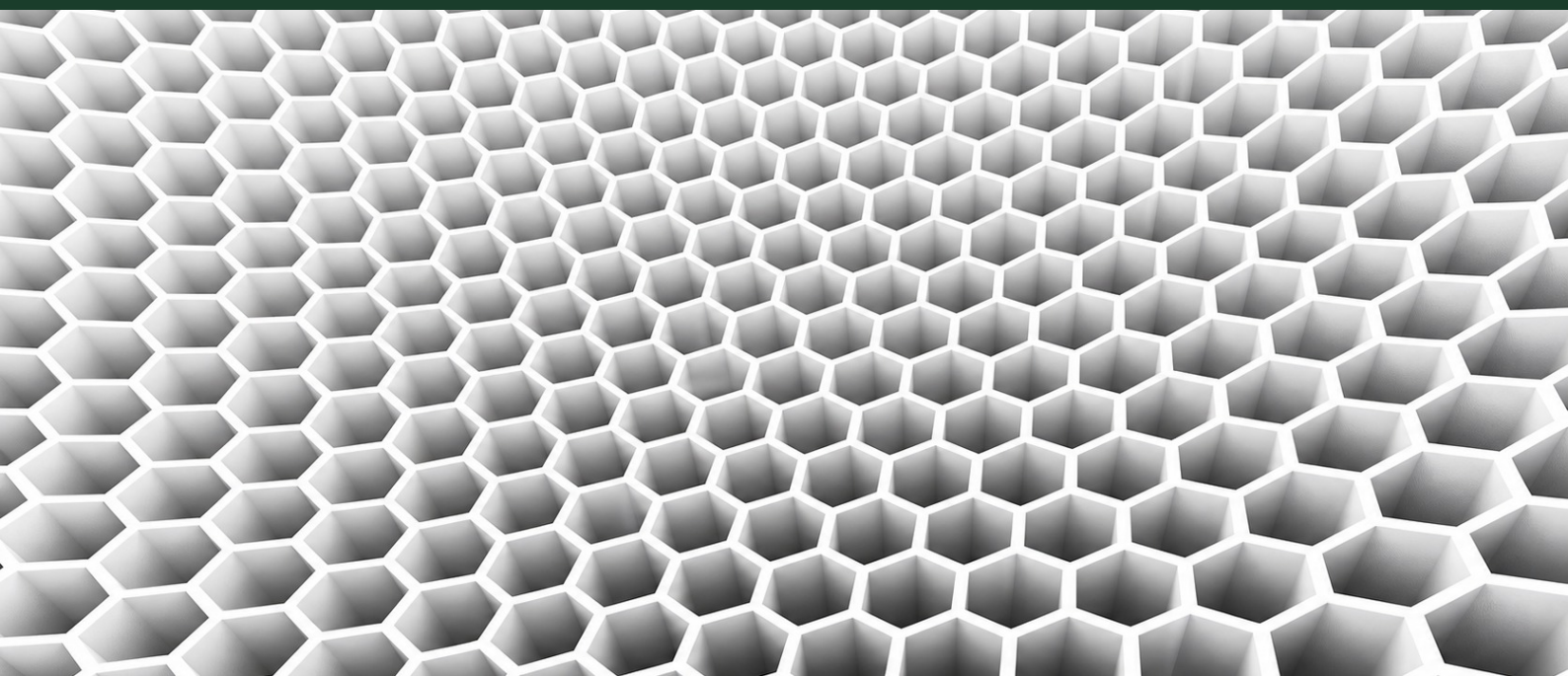
BEYOND AGENT TEAMS

Multi-Agent Systems Need an Operating Control Plane — Not More Intelligent Agents

The evidence for governing hand-offs, state, authority and recovery across enterprise agent workflows

Giovanni Leonardi

August 2024



Multi-Agent Systems Need an Operating Control Plane — Not More Intelligent Agents

Programme · No. 267
Originally written August 2024 | Re-edited and re-rendered for the WhereBeyond Edition,

"Multi-agent orchestration requires an operating control plane for meaning, state, authority and recovery."

1. The demonstration worked; the programme did not

A programme team asks a chain of six agents to assess proposed scope changes. One extracts the request, one retrieves requirements, one estimates delivery impact, one identifies risks, one challenges the analysis, and one prepares a recommendation. In the demonstration, the sequence produces a polished answer in under four minutes.

During live use, the same chain becomes unreliable. A change marked "subject to architecture review" is treated as approved after the qualification disappears from an intermediate summary. The estimating agent assumes a dependency has been resolved because no dependency is listed. The challenge agent critiques the recommendation but cannot see the source record. The final agent reconciles the disagreement by choosing the most frequently repeated claim.

No individual agent has obviously failed. The system has failed as an enterprise process.

This pattern is becoming visible as organisations move beyond single assistants and experiment with tool-using, role-based agent workflows in 2024. The prevailing response is to improve prompts, assign clearer personas or introduce another supervisory agent. Those actions may improve local performance. They do not solve the underlying problem.

The evidence points to a more demanding conclusion. Multi-agent orchestration requires an operating control plane for meaning, state, authority and recovery. Without it, additional agents multiply plausible outputs while accountability dissolves between them.

2. Why multi-agent designs are attractive

The technical argument for multiple agents is strong. Complex tasks become difficult to manage inside one long instruction. Decomposition allows retrieval, analysis, critique and action to be tested separately. Specialised agents can use different context, tools and constraints. A challenger can expose weaknesses in a first answer. Components can be replaced without rebuilding the entire chain.

In enterprise delivery, that modularity promises three benefits:

- Separation of concerns: evidence gathering is not confused with approval or execution.
- Independent challenge: a second reasoning path can test assumptions and omissions.
- Scalable workflow: tasks can run sequentially or in parallel without constant human coordination.

Yet these benefits appear only when the interfaces between agents carry the distinctions the organisation depends upon. A programme has facts, assumptions, options, decisions, permissions and unresolved questions. If they all travel as prose, specialised agents inherit an undifferentiated narrative. The architecture is modular in code and monolithic in meaning.

3. What the early evidence is showing

Across current enterprise experiments, four failure mechanisms recur.

3.1 Context degrades at each hand-off

Agents tend to compress. Compression is useful, but it can remove provenance, qualification and uncertainty. “Legal review requested; no response received” becomes “legal review pending,” then “pending approval,” then “approval expected.” Each phrase is plausible. The final meaning is materially different from the source.

This is not merely a hallucination problem. It is lossy transmission. Even accurate agents can create an inaccurate chain when each receives a transformed account rather than the evidence itself.

3.2 Shared state is neither shared nor stable

Multiple agents may act on different versions of the same case. One retrieves the latest delivery plan; another uses a cached extract; a third writes an updated assumption that the others never see. The final result looks coherent because language models are good at

producing coherence. Coherence is not proof that the underlying state was consistent.

3.3 Role labels are mistaken for authority

Calling an agent "approver," "risk lead" or "programme controller" does not confer organisational authority. A generated conclusion may inform a decision; it cannot acquire formal standing through a prompt. Where systems blur recommendation and approval, responsibility shifts into implementation detail without an explicit delegation.

3.4 Consensus disguises correlated error

Adding agents does not automatically add independent judgement. Agents may share the same source, model, prompt pattern and missing information. Majority agreement can therefore repeat one error several times. A supervisory agent that resolves conflict by selecting consensus may make the system more confident without making it more correct.

> *Agent count is not a measure of independent assurance when the agents share the same evidence, assumptions and blind spots.*

4. A composite delivery case

Consider an enterprise portfolio using a multi-agent workflow to screen 240 monthly change requests. The pilot reports that 84 per cent of recommendations match the eventual decision of the change authority. Average preparation time falls from 48 minutes to 11.

The headline appears persuasive until the cases are segmented.

Of the 240 requests, 176 are routine date or wording changes. Agreement on these cases is 94 per cent. The remaining 64 alter cost, architecture, supplier obligation or regulatory interpretation. Agreement there is 58 per cent. In 17 material cases, the workflow produces a recommendation without a complete dependency record. Human reviewers catch 13; four proceed to the agenda with false completeness.

A review finds that the retrieval agent correctly marked missing evidence. The planning agent converted blanks into "no impact identified." The risk agent was instructed to challenge stated risks, not missing evidence. The final agent interpreted the absence of disagreement as confirmation.

The intervention did save preparation time. It also changed the failure mode: incompleteness became invisible earlier and travelled faster.

The team tries three remedies.

1. More detailed prompts reduce some wording errors but make the chain brittle when document formats change.
2. A supervisor agent improves consistency but repeats the same missing-evidence assumption because it sees only intermediate summaries.
3. A mandatory evidence-and-state contract preserves source status, prevents blanks from becoming negative findings, and routes incomplete material to a human owner.

Only the third remedy changes the mechanism. After six weeks, material cases with incomplete dependencies fall from 17 to three; all three are visibly held rather than silently progressed. Preparation time settles at 16 minutes rather than the pilot's 11, because the control adds work. That is the more credible result: lower speed than the demonstration, stronger integrity than the original process.

5. The control plane the enterprise needs

A control plane is not another agent. It is the set of operating rules and technical services that govern how agents exchange evidence, act on state and escalate consequence.

Control plane element	Purpose	Required design
Handoff contract	Preserve meaning between agents	Typed fields for fact, inference, status, source and confidence
State authority	Establish which version governs	Timestamped record, version owner and conflict rule
Decision rights	Separate advice from action	Permitted actions, value limits, prohibitions and human owner
Observability	Reconstruct the chain	Inputs, transformations, tool calls, exceptions and final action
Exception routing	Stop unsafe continuation	Conditions for pause, evidence request, human decision or termination
Recovery	Limit consequences	Reversal method, state restoration and affected-party response

5.1 Handoff contracts

Each interface should define what may be passed, what must remain unchanged and what the receiving agent may infer. High-consequence fields should not be buried in narrative summaries.

A practical contract distinguishes:

- source fact from agent inference;
- confirmed status from expected status;
- “not applicable” from “unknown”;
- advice from authorised instruction;
- confidence from consequence.

The purpose is not to eliminate natural language. It is to keep organisationally significant meaning from being flattened by it.

5.2 Authoritative state

One record must govern the workflow at a given moment. Agents can maintain working memory, but changes to shared state require a defined writer, version and validation rule. When agents disagree about state, the system should expose the conflict rather than silently merge it.

This matters especially when agents operate in parallel. Speed gained through parallel work is easily lost if outputs are reconciled against different versions of the case.

5.3 Decision rights and action envelopes

Every agent needs an explicit action envelope: which systems it may access, which decisions it may support, which actions it may execute, and which conditions require human authority.

The envelope should be narrowest where effects are irreversible, external, financial or regulated. An agent may draft a change recommendation while being prohibited from updating the approved baseline. It may identify a supplier exposure while being prohibited from contacting the supplier. These distinctions should be enforced in permissions and workflow rules, not left as polite instructions.

5.4 Chain-level observability

Component logs answer whether an agent ran. Programme evidence must answer why the overall action occurred. Retain the source reference, relevant state, agent version,

intermediate transformation, tool call, exception and human intervention.

The goal is reconstructability, not indiscriminate retention. The organisation needs enough evidence to replay a consequential case and identify where meaning or authority changed.

5.5 Designed exceptions

A mature workflow expects missing evidence, disagreement and contradiction. It defines what happens before those conditions arise.

Exceptions should be based on consequence and uncertainty rather than confidence alone:

- missing mandatory evidence pauses the chain;
- conflicting authoritative sources route to the state owner;
- agent disagreement on a material case triggers independent review;
- any proposed irreversible action requires named human authority;
- repeated exceptions of one type trigger workflow redesign.

The exception route is not a defect around the automated process. It is part of the process.

6. Why common alternatives fall short

The strongest case against a control plane is that early agent systems are experimental. Heavy governance may freeze a fast-moving design before teams understand it. That concern is valid. A large central committee, exhaustive schema and fixed model choice would turn learning into paperwork.

The answer is proportional control, not absence of control. Exploration that cannot alter records, money, commitments or external communications can operate with lighter contracts. As authority and consequence increase, structure must increase with them.

Three other alternatives are inadequate.

6.1 “Keep a human in the loop”

Human review works only when the reviewer has time, evidence, competence and power to refuse. If one person receives hundreds of fluent recommendations with no visible provenance, the human becomes a confirmation step. Meaningful review requires exception concentration and reconstructable evidence.

6.2 "Use a stronger supervisor"

A supervisor agent can route tasks and check format. It cannot manufacture missing evidence, confer authority or guarantee independence from agents that share its model and context. Supervision is a function within the control plane, not a substitute for it.

6.3 "Evaluate each agent separately"

Component evaluation is necessary. It does not measure interaction failure. The enterprise must also test full paths, incomplete cases, conflicting sources, tool failure, stale state and recovery. The chain is a separate object of assurance.

7. The recommended operating model

Organisations should establish a joint agent orchestration authority for each material workflow. This need not be a permanent committee. It is a named partnership between the programme decision owner, technical product owner and relevant control owner.

Before deployment, that partnership should approve five artefacts:

1. Decision topology: where facts become inferences, recommendations become actions, and human accountability attaches.
2. Handoff catalogue: required fields, provenance, permitted transformations and failure behaviour at each interface.
3. State model: authoritative records, version rules and conflict handling.
4. Action envelope: permissions, limits, prohibitions and escalation thresholds for each agent.
5. Evidence and recovery plan: logs, replay tests, stop mechanism, reversal and incident ownership.

Deployment gates should test the workflow with more than happy-path examples. At minimum, run a complete case, an incomplete case, a contradictory case, a stale-state case and a tool-failure case. For material actions, demonstrate that the chain stops safely and that an investigator can reconstruct why.

Once live, measure the system at chain level:

- completion rate by case type;
- exception and escalation rate;
- unsupported inference rate;

- state conflict rate;
- human override and reason;
- downstream rework;
- time and cost including review;
- material incidents and near misses.

These measures reveal whether orchestration improves the programme rather than merely producing faster text.

8. A decision rule for scale

A multi-agent workflow is ready to scale when three conditions hold simultaneously.

First, it produces a sustained operating benefit after the cost of review, control and maintenance. Second, material decisions remain inside explicit authority envelopes with tested exception routes. Third, consequential cases are reconstructable from source to action.

If any one condition is missing, the appropriate response is to redesign or contain the workflow—not to add another intelligent component.

The present enthusiasm for agent teams is understandable. Decomposition can make complex work more tractable, and early demonstrations show genuine promise. But enterprise delivery has never failed for lack of boxes on a process diagram. It fails where meaning changes between boxes, where nobody owns the state, and where activity crosses into authority without a decision.

Multi-agent orchestration will become dependable when organisations stop treating it as a conversation among clever assistants and start treating it as an operating system for consequential work.