Work as a Packaged Item: North-Star Adjudication for Multi-Agent Systems

Abstract

Multi-agent systems are good at *doing* and bad at *finishing*. An agent, or a chain of agents, will report success on a task it did not complete — not from dishonesty but because "done" is, by default, self-declared: the agent that did the work is the same party that judges whether the work satisfies the goal. We argue that reliable autonomous work requires two structural moves the current agent stack mostly lacks: (1) **reifying work as a first-class, durable, routable item** — a *work packet* carrying intent, resources, acceptance criteria, budget, and governance tier — rather than an ephemeral prompt; and (2) **adjudicating completion against that packet's acceptance criteria by a party other than the executing agent**, checking emitted evidence rather than trusting a self-report. We call the second move **north-star adjudication**. We ground the argument in production data from ARQERA — **237 recorded agent runs** whose real status distribution (43% succeeded, **37.5% escalated to a human**, 18.6% failed) exposes exactly the "confidently-done-but-not-done" gap self-declaration hides — and in the coordination substrate that carries the packets (**1,213 tasks and 4,131 messages across 31 registered agents; ~179,000 governed inter-agent envelopes**). We describe a completion adjudicator that verifies acceptance criteria against the append-only evidence ledger, and we are explicit about what is wired versus dark: the adjudicator and the packet-based dispatch path exist and are reviewed, but are enabled behind a flag pending live validation on a real packet. The contribution is the *design pattern* — packet + external adjudication — and the honest production evidence for why the pattern is needed.

1. The finishing problem

Ask a single agent to complete a multi-step task and it will often stop early and declare success. Chain several agents and the failure compounds: each hands off a partial result the next accepts as complete. The root is not capability; it is *epistemics*. In the default agent stack, the answer to "is the work done?" is produced by the agent that did the work. Self-grading is a known-bad evaluation design in every other setting; we have imported it into the core of autonomous execution and are surprised when agents mark their own incomplete work as complete.

Two things are missing:

1. **The work is not a durable object.** It exists as a prompt and a transient run. There is no first-class record of *what was asked, with what resources, to what acceptance criteria, under what budget and governance* — so there is nothing concrete for a completion check to check *against*.
2. **Completion is not adjudicated.** Even where acceptance criteria exist, nothing independent verifies them against what the agent actually produced.

This paper proposes fixing both, and reports production evidence for why it matters.

2. The production evidence: 37.5% escalation is the tell

ARQERA records agent runs as durable rows. Across **237 runs**, the status distribution is:

Status	Count	Share
succeeded	102	43.0%
escalated (to a human)	**89**	**37.5%**
failed	44	18.6%
blocked	2	0.8%

The **37.5% escalation rate** is the empirical fingerprint of the finishing problem. These are runs where the system correctly declined to self-declare success and instead routed the decision to a human — which is the *right* behaviour, but it also means **more than a third of autonomous runs could not be adjudicated complete by the agent alone.** A system that trusted self-declaration would have marked many of these "done." Historical ledger forensics on the same platform are starker still: in an earlier, larger sample the escalation rate exceeded 80%, and of escalations that reached a human, the overwhelming majority were ultimately rejected — evidence that ungoverned self-declaration and mis-tiered actions produce a flood of low-value "is this done / may I proceed" checkpoints. The lesson is consistent: **left to itself, a multi-agent system cannot reliably tell whether it has finished, and either over-claims completion or over-escalates.** Neither is acceptable; both point at the same missing primitive — an *external* completion check against a *concrete* specification.

---

3. Move 1: reify work as a packet

The first move is to make work a first-class, durable, routable object — a **work packet** — rather than a prompt. A packet carries:

- **Intent** — the goal, in the requester's terms.
- **@resources** — the specific accounts, connectors, and data the work may touch (bound, not implicit).
- **Acceptance criteria** — the checkable conditions that define "done" (e.g. "a non-empty table exists satisfying X"; "the invoice was sent and a receipt recorded").
- **Budget** — the spend/step ceiling.
- **Governance tier** — the authority class of the actions the work may take (auto / soft / hard).
- **Prompt + context** — the executable instruction and the assembled situation.

Reifying work this way makes it *routable* (it can be dispatched to the best-fit agent, workforce, or pod by capability), *auditable* (the packet is the durable record of what was asked), and — critically for move 2 — *checkable* (the acceptance criteria are the specification a completion adjudicator verifies against). Work becomes an item you can route, meter, govern, and prove — the way a logistics network treats a parcel — rather than a fire-and-forget instruction.

ARQERA carries packets over a coordination substrate that is live and load-bearing: **1,213 coordination tasks, 4,131 inter-agent messages, and 31 registered agents**, with agent-to-agent communication flowing as **~179,000 governed envelopes** (every inter-agent message is itself a governed, evidence-emitting action, so delegation is

auditable end-to-end). The dispatch layer selects an executing agent by capability, availability, and workload; the messaging layer carries `DELEGATE` / `INFORM` semantics; a lifecycle layer drives each agent through *arrive → brief → equip → confirm → release → debrief*. The pieces of a real swarm exist; the missing piece was the adjudicated close.

---

4. Move 2: north-star adjudication

> **North-star adjudication.** Completion is decided by verifying the work packet's acceptance criteria against the *emitted evidence*, by a party other than the executing agent. "Done" is adjudicated, never self-declared.

Concretely, a completion adjudicator takes a run's packet and its acceptance criteria and checks each criterion against the append-only evidence ledger — the same independent, hash-chained record described in the companion BYOK paper. The verdict is *met* or *not-met with the specific failed criteria*, and it is produced by the adjudicator, not the agent. This mirrors a pattern already proven elsewhere in the platform: readiness for a code merge is *not* something a worker may self-assert; a substrate-side adjudicator fetches the ground-truth state (from the version-control provider, the CI results, the required checks on the exact commit) and emits the canonical verdict — a worker may only *submit a claim*, and the adjudicator decides. North-star adjudication generalises that discipline from "is this code ready to merge" to "is this work done."

The name encodes a second discipline the design insists on: the packet's objective is the *north star*, re-injected at each step so the executing agent cannot drift — the common autonomous-agent failure mode where an agent, blocked on a sub-task, retreats, forgets the original goal, marks the sub-task done, and thereby fails the real ask. Anchoring the objective and adjudicating against it (rather than against the agent's own moving account of what it was doing) is what makes "the swarm finished the job" a checkable claim rather than a hope.

---

5. Honesty about state: wired vs dark

Research on one's own production system must be explicit about what is *running* versus what is *built but not yet enabled*. In ARQERA:

- The coordination substrate (tasks, messages, agents, governed envelopes) is **live** — the numbers in §3 are production counts.
- Agent runs as durable, status-carrying objects are **live** — the distribution in §2 is production data.
- A packet-based dispatch path (routing multi-capability work through the capability-selecting dispatch engine and governed messaging rather than a single-agent run) and the completion adjudicator are **built and reviewed but gated behind a flag** (default off), pending live validation against a real packet end-to-end. They are *dark plumbing*, not yet load-bearing.

We consider this distinction essential, not a caveat to bury. The *pattern* — reify work as a packet, adjudicate completion externally against its criteria — is the contribution and is supported by the production evidence for *why* it is needed (§2). Whether ARQERA's specific implementation delivers the guarantee is exactly the thing

that must be shown by adjudicating a real packet to *done*, not self-declared — and the design's own principle forbids us from claiming otherwise until it is.

---

6. Discussion

The finishing problem is where "agent" and "system" meet. An agent can be arbitrarily capable and still fail to finish if the *system* provides no durable specification to finish against and no independent party to confirm it finished. The two moves — packet and external adjudication — are deliberately *systems* moves, not model moves: they do not make the agent smarter; they change what the agent is accountable to. This is the same shape as the wieldability argument (a companion paper): the binding constraint on autonomous work is a property of the *system*, and the fix is a *surface* (here, the packet and the adjudicator), not a better model.

The pattern connects to alignment and evaluation research in a specific way: north-star adjudication is *process-level* outcome verification. It does not verify the agent's reasoning; it verifies, against an independent ledger, that the *stated acceptance criteria* were met. This is weaker than verifying the agent did the right thing for the right reason, and stronger than trusting a self-report — a pragmatic middle that scales, because acceptance-criteria-against-evidence is cheap to check where "was the reasoning sound" is not.

Limitations

The central honesty caveat is §5: the adjudicator and packet-dispatch path are dark, so the paper demonstrates the *need* for the pattern (via the 37.5% escalation data and the historical forensics) more than it demonstrates the pattern *delivering* a completion guarantee at scale. The 237-run sample is modest and platform-specific; the escalation rate is a real signal but not a population constant. Acceptance-criteria adjudication is only as good as the criteria: a packet with weak or unfalsifiable criteria can be adjudicated "done" while failing the true intent — pushing the hard problem to criteria authoring, which the paper does not solve. Finally, external adjudication adds latency and a new dependency (the ledger and the adjudicator) to every close; whether that cost is justified depends on the value of a false-"done", which is high for governed work and low for trivial tasks — the pattern should be applied selectively, not universally.

---

7. Conclusion

Multi-agent systems finish unreliably because completion is self-declared against nothing durable. Making work a first-class packet (intent, resources, acceptance criteria, budget, governance) gives completion something concrete to be checked against; adjudicating that check against an independent evidence ledger, by a party other than the executing agent, makes "done" a verdict rather than a claim. The production evidence — a 37.5% human-escalation rate across 237 runs, a live coordination substrate of thousands of governed tasks and messages, and historical forensics of over-escalation under self-declaration — shows the finishing problem is real and structural. The design that addresses it is a systems design, and we hold ourselves to its own standard: the guarantee is not proven until a real packet is adjudicated to done, and we say so.

Appendix A — Headline figures

Figure	Value
Recorded agent runs	237
— succeeded	102 (43.0%)
— **escalated to human**	**89 (37.5%)**
— failed	44 (18.6%)
— blocked	2 (0.8%)
Coordination tasks / messages / agents	1,213 / 4,131 / 31
Governed inter-agent envelopes	~179,000
Completion adjudicator + packet dispatch	built, reviewed, **flag-gated (dark)**

All figures measured live on the production platform, 23 July 2026.